LiU-ITN-TEK-A--17/043--SE

Multi-Touch Interfaces for Public Exploration and Navigation in Astronomical Visualizations

Jonathan Bosson

2017-09-05

Department of Science and Technology
Linköping University
SE-601 74 Norrköping, Sweden

Institutionen för teknik och naturvetenskap
Linköpings universitet
601 74 Norrköping

LiU-ITN-TEK-A--17/043--SE

Multi-Touch Interfaces for Public Exploration and Navigation in Astronomical Visualizations

Examensarbete utfört i Medieteknik
vid Tekniska högskolan vid
Linköpings universitet

Jonathan Bosson

Handledare Emil Axelsson
Examinator Anders Ynnerman

Norrköping 2017-09-05

Acknowledgements

I would like to thank the Scientific Computing and Imaging (SCI) Institute at the University of Utah and the OpenSpace developer group for the opportunity to complete this master thesis project, especially my supervisors Alexander Bock, Emil Axelsson and Dr. Charles Hansen for insights and guidance regarding the thesis. Addidionally, I would like to thank my examiner Anders Ynnerman for arranging the collaboration with the SCI institute making my thesis project possible.

Jonathan Bosson

Abstract

This thesis describes a project in Media Technology that was carried out at the Scientific Computing and Imaging Institute at the University of Utah and the Department of Science and Technology at Linköping university.

OpenSpace is an interactive data visualization software system that portrays the entire known universe in a 3D digital simulation. Interacting with the application currently involves mouse and keyboard input as well as expert knowledge of the architecture of the system and its graphical user interface. Although navigation interaction for exploratory purposes are mostly possible through mouse interaction it requires explanations and is locked to a single person, which prohibits OpenSpace to be displayed effectively in public exhibitions.

Research has been shown that using large tangible touch surfaces with a multi-touch navigation interface is more engaging to users than mouse and keyboard as well as enhances the understanding of navigation control, thus decreasing the learning time of the systems user interface. This thesis shows that combining a velocity-based interaction model together with a screen-space direct-manipulation formulation produces a user-friendly interface. Giving the user precise control of objects and efficient travels in between in the vastness of space. This thesis presents the work of integrating a multi-touch navigation interface with a combined formulation of velocity-based interaction and screen-space direct-manipulation into the software framework OpenSpace.

Contents

1 Introduction **1**

1.1 OpenSpace Project 1
1.2 Motivation 2
1.3 Aim 3
1.4 Research Questions 3
1.5 Delimitations 4
1.6 Method 4

2 Related Work **7**

2.1 Touch Input 7
2.2 Beginning of Direct-Manipulation Formulations 7
2.3 Using Screen-Space 9
2.4 DS3 and Separation of DOF 12

3 Theory **15**

3.1 Choices in DOF Separation 15
3.2 Creating the Minimize Function and its Gradient 17
3.3 Finding Selected Objects 19

4 Implementation **21**

4.1 Decoupling the Input 21
4.2 Putting Markers in Screen-Space 22
4.3 Determine Type of Interaction Scheme 22
4.4 Interaction Types and Interpretation 24
4.5 Velocity-based Method 25

5 Results **27**

5.1 Discussion 28

6 Conclusions and Future Work **30**

6.1 Conclusions 30

6.2 Future Work 31

References **32**

List of Figures

1.1 3D Astronomical Visualization software OpenSpace using multi-touch to control the view in six degrees of freedom (DOF). 1
1.2 Software architecture of the OpenSpace touch module that allows a user to interact with the 3D astrovisualization software through multi-touch devices. 6

2.1 Formulation of Depth-Separated Screen-Space with the help of Z-technique. 1d stands for one active contact point which is in direct contact to the virtual object, 1d+1i is thus with an additional indirect contact point to allow control over an additional DOF. 13
2.2 Comparison between the existing approaches' control of the 6DOF, based on the number of contact points and their directness (e.g., 2d+1i: two contact points directly touch the virtual object with one indirect finger). . . . 13

3.1 OpenSpace's DOF separation for direct-manipulation control. With the help of one, two or more active contact point that touches the virtual object. 17

4.1 Alpha fall-off function dependant on the choice of thickness and transperancy cap 23

5.1 Shows different interactions in OpenSpace. (a) and (b) has two contact point interacting with the Moon and Earth respectively to control the camera. (d) illustrates the gesture to control the roll DOF. 27
5.2 The red and blue lines are the LMA minimization paths for each iteration step for respective contact point for this particular interaction. Large circles denotes the contact points initial location and the smaller ones the goal position. 28

1. Introduction

This master's thesis has been carried out together with OpenSpace, the Scientific Computing and Imaging Institute at the University of Utah and the Department of Science and Technology at Linköping University. This chapter presents the partners in the OpenSpace project as well as the motivations and goals for the thesis.

Figure 1.1: 3D Astronomical Visualization software OpenSpace using multi-touch to control the view in six degrees of freedom (DOF).

1.1 OpenSpace Project

OpenSpace [1] is an open-source astronomical visualization framework for interaction with large dataset. Partially supported by NASA, it is designed to visualize the entire known universe and the ongoing efforts of investigating the cosmos. The software system originates from an academic collaboration between Sweden's Linköping University (LiU), American Museum of Natural History (AMNH) and NASA Goddard's Community

Coordinated Modeling Center (CCMC). It was developed several years ago with its initial purpose to model space weather forecasting. Continued efforts resulted in visualizations of NASA's New Horizons mission to Pluto and ESA's Rosetta mission. Recent NASA funding allowed the project to be extended in collaboration partners and areas of focus. With the University of Utah's Scientific Computing and Imaging (SCI) Institute, New York University's Tandon School of Engineering, multiple informal science institutions across the United States, as well as several international vendors added to the list of collaborators. Currently OpenSpace focuses on visualizing globe browsing tecniques and to visualize dynamic simulations using interactive volumetric rendering to promote communication of research in astrophysics. By utilizing NASA's SPICE observational geometry system with its Planetary Data Service (PDS) can space missions be visualized to evoke interest and portray how science is gathered.

1.2 Motivation

Touch-based interaction has during the last few years, become a more and more common way to interact with technology. Touch screen technology has advanced quickly and become the standard in today's smartphones. The interface control and the intuitive feel is naturally highly dependant on the method that is implemented. A common approach is to use the velocity in which the control points are moving across the touch surface. Gestures are then used to interpret different types of interactions and at what speed those interactions are desired. The challenge with this method is to design the gestures to be well defined such that they do not overlap as well as keeping them intuitive to users, which can easily differ from person to person. Basing the interaction on the velocity of fingers also enforces a mapping between the movements on the touch surface to interaction in the application which the user has to account for.

Another approach is to design a type of direct-manipulation, which aims to give the user the feeling of interacting with real objects. As the user moves their fingers along the touch surface, virtual objects rotate, translate and scales appropriately such that the object or area always remains underneath the same fingertip. This technique, often called Rotate-Scale-Translate (RST) interaction, has essentially become the *de-facto* standard in 2D touch environments to provide an intuitive and controllable mapping between points in an object's local space and points in screen space without the need for any explicit gesture processing. Higher dimensional manipulation faces a challenge since it involves the control of an increasing number of degrees of freedoms (DOF). With an increased

complexity it becomes exceedingly difficult for the user to achieve the desired result. Extending the methodology to 3D environments and higher dimensional use cases have been researched through various methods, the most prevalent formulations being Screen-Space (SS) by Reisman et al. [2], Sticky tools by Hancock et al. [3] and Depth-Separated Screen-Space (DS3) by Martinet et al. [4]

Astronomical visualizations has long been an interesting application for touch-based interaction. However, due to the scale of the solar system compared to its celestial bodies, any existing object becomes exceedingly small. A direct-manipulation solution alone becomes impossible in an application like this since its formulation require 3D points in the scene to constrain the manipulation and such points cannot be traced in empty space. For further reading about this problem, see Chapter 3. Given a solution, using touch screens to interact with OpenSpace's 3D visualizations raises as an interesting application for research purposes and especially to public exhibitions with the aim to educate, stir interest and make space science more accessible to the public.

1.3 Aim

The aim of this thesis is to investigate the concept of a user-intuitive multi-touch navigation interface system to explore real-time astronomical visualizations. This is achieved by intertwining a Screen-Space direct-manipulation formulation together with a velocity-based model seamlessly for exploration in empty space as well as close to the surface of celestial bodies.

Any issue addressing complex user-interface designs is based upon decisions between freedom and simplicity. Simplicity and constraints often go hand in hand, especially when it comes to interfaces for 3D environments. Three dimensions means control over six degrees of freedom; spatial position (x, y, z) as well as orientation often described through *pitch*, *yaw* and *roll*. Although evidence exists that touch provides a higher-bandwith input, such as the study by Jacob et al. [5], allowing two dimensional data produced from touch screens to control over all degrees of freedom requires efforts in classification of DOFs.

1.4 Research Questions

The research questions of this thesis are the following:

- What constraints should be set to the touch-interface such that the user finds the interface intuitive without compromising the ability to manipulate all six degrees of freedom for astronomical visualizations?

- How can an interface be designed to seamlessly swap between two radically different interaction modes?
- What design choices must be considered to ensure that interactions are decoupled from frame time such that different hardware systems generate the same result?
- What type of screen-space direct-manipulation evaluation generates good results while handling numerical computation limitations?

1.5 Delimitations

The delimitations for the thesis are specified below.

- The touch framework TUIO proved to support the largest number of operating systems (OS) and has thus been used as the touch protocol to which OpenSpace listens. A bridge application that translates Windows Touch events into TUIO was created to ensure interaction with the most popular OS on public exhibition touch tables.
- Although numerous different curve-fitting solvers exist, Levenberg-Marquardt algorithm has been used to solve a non-linear least-square minimization function.
- Mitigation methods of rotational extrema to resolve issues such as rotational extrema or ambiguous rotations have not been implemented in this thesis.
- The touch interface has been developed such that it collaborates well with the mouse and keyboard interface already in place. The most notable restriction being the requirement to keep a *node*, ie. celestial body, in focus such that camera movements are relative to the node's position.
- Two types of user tests were conducted. A small group of returning participants at different iterations of the interface guided by their feedback, as well as two sessions with a larger group of people between the ages of 20-50 years old.

1.6 Method

The agile development framework Scrum [6] was employed to most areas in the development process with the exceptions of daily meetings, since the thesis was held by a single person. Each week had its specific focus much like *Sprints*, with a backlog with features partitioned into *Stories* and *Tasks*. Weekly meetings together with the code master of OpenSpace and thesis handler functioned as sprint reviews as well as planning meetings

for the coming week. Additionally, an weekly entry were posted to OpenSpace's blog website. A strength of developing alone was that a lot of time consuming procedures in attempts to increase communication between teams could be exempt from the process. However, the lack of multiple people with expert knowledge of the thesis caused the direction to become unclear, at times, in terms of minor system architectual decisions.

Automatic build testing on all three major operating system and code analysis were done by Jenkins for each commit submitted to the git branch. This allowed for instant notification when faulty code had been submitted and logs to trace where the error appeared. Interface design decisions were guided by the feedback of two types of user tests. The first type had a large sample of people, all seeing the system for the first time while the second type was performed by inhouse developers that returned throughout the project period to iterate the feedback after changes. The first user test type were done two times and helped to evaluate how intuitive the interface was depending on prior knowledge and expectations of touch screens. The reoccuring user tests were more helpful as quality assurance (QA) tests as it was more focused on processes and procedures rather than conducting actual tests. This ensured that bugs were identified and removed as well as giving in-depth feedback to the overall interface system.

OpenSpace is a modular simulation system such that it can load different functionalities depending on the use case. Multi-touch interaction could thus be developed by introducing a new module that handled everything related to touch input. The touch module introduces four new classes as well as the module class as seen in figure 1.2. The *TuioEar* is the client-side of the TUIO protocol, and listens to input sent through a TCP/IP or UDP port. Each frame the module class asks the client for new input and sends the data further to the *TouchMarker* and *TouchInteraction*. The *TouchMarker* simply renders the contact points with the chosen appearance while the *TouchInteraction* is the kernel of the multi-touch interactivity wherein the camera is manipulated to a new state based on the received input. The final class is the LMA solver *levmarq* that, if active, computes the gradient descent to help give the best-fit camera parameters for the new state.

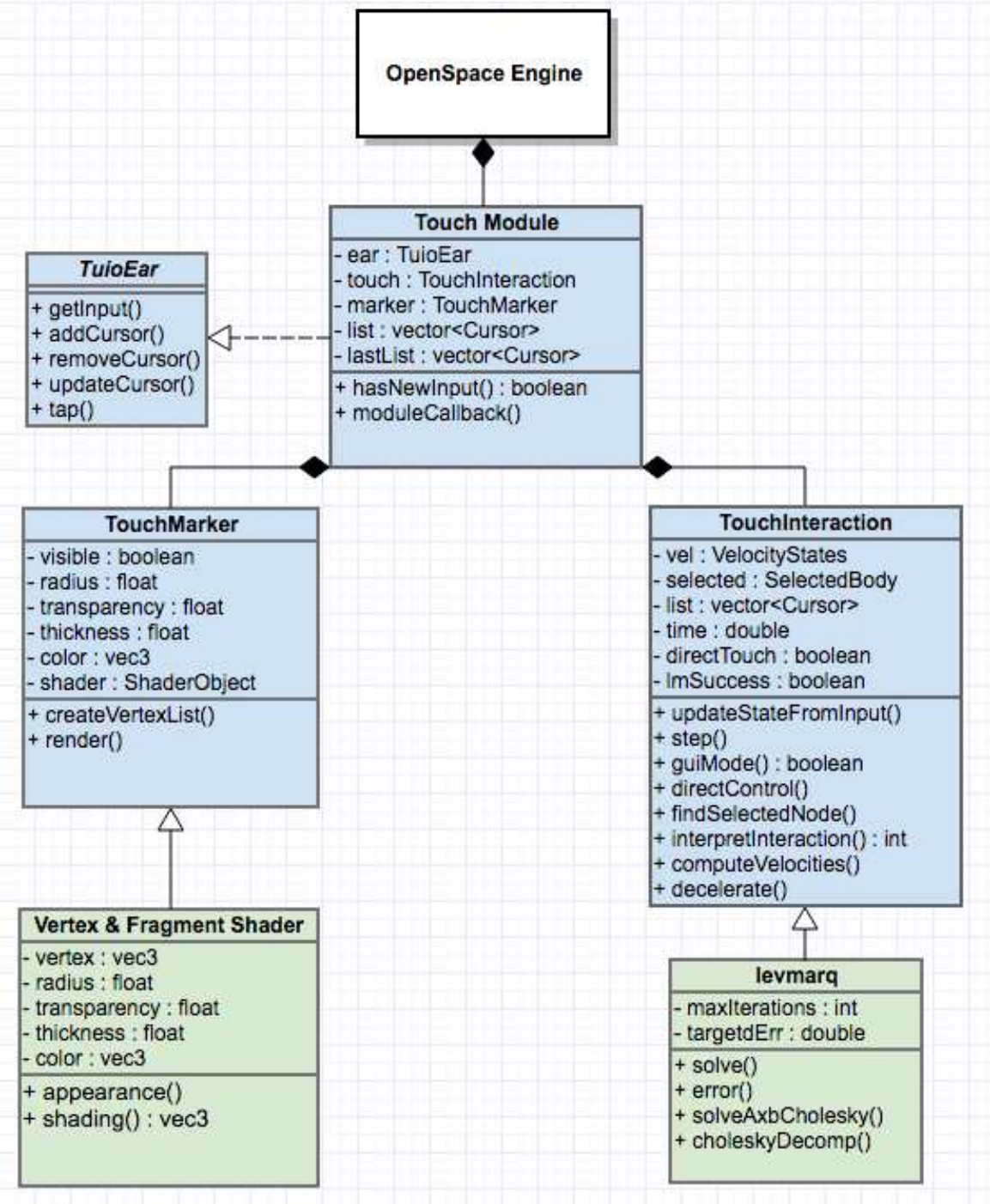


Figure 1.2: Software architecture of the OpenSpace touch module that allows a user to interact with the 3D astrovisualization software through multi-touch devices.

2. Related Work

This chapter describes previous work which is relevant to this thesis, most notably the history and current status of direct-manipulation formulations and how such methods achieve user-friendly interfaces in 3D applications.

2.1 Touch Input

Touch frameworks encode the hardware contact data to a standardized input such that build their projects without specific hardware dependencies, the by far most used one being Windows Touch protocol. However, it comes with the dependency of running a Windows operating system on the used computer which might not be desired. Other prevalent frameworks are mtdev or, the one used in OpenSpace, TUIO. TUIO is developed by Kaltenbrunner et al. [7] and uses a client-server transmission protocol to transfer the control data from hardware to software. It is designed as an abstraction layer for tangible multitouch surfaces that allows the protocol to be agnostic of the operating system. Instead, the developed application becomes a client listening to either a TCP/IP or UDP port that the server is sending its data through. Due to this the server, and thus the user, can be far away from the client application and has freedom to choose whichever interactive surface system that is desired. The user can interact with the client through a smartphone, tablet or large touch table system running any operating system.

2.2 Beginning of Direct-Manipulation Formulations

One of the first papers on a direct-manipulation method was proposed 1992 by Gleicher et al. [8] focused on moving the camera rather than any virtual object. The problem Gleicher et al. aimed to solve was the inflexibility of different camera model's parametrizations. The underlying model for camera formulations is the perspective projection, through which each 3D view is completely specified with the center of projection, the view plane as well as the clipping volume. However, what parameters are used to control the camera

can vary. The need for specialized control arises frequently but deriving those transformations pose a difficult problem. Gleicher et al. offered a solution to this issue by having the interaction constrained such that contact points remain inside the image. The key difference from previous methods was that Gleicher et al. formulated the problem to solve for the time derivatives of the camera parameters rather than the parameters directly. Equation 2.1 describes the relation between a 3D point $\mathbf{x}$ and the projected 2D coordinates $\mathbf{p}$ on the screen

$$\mathbf{p} = h(V\mathbf{x}), \tag{2.1}$$

where V is a homogeneous matrix representing the combined projection and viewing transformations. h is the function that converts the result into 2D image coordinates by normalizing the x and y value with its fourth component as

$$h(\mathbf{x}) = \begin{bmatrix} \frac{x}{w} & \frac{y}{w} \end{bmatrix}. \tag{2.2}$$

The V matrix is in this case the product of several matrices, each a function of one or more of the camera parameters $\mathbf{q}$. This allows the image point $\mathbf{p}$ to be described as a non-linear function of the parameters $\mathbf{q}$ and 3D point $\mathbf{x}$, assuming $\mathbf{x}$ is fixed. The 2D point velocity $\dot{\mathbf{p}}$ can thus be computed as follows:

$$\dot{\mathbf{p}} = h'(V\mathbf{x})(\frac{\delta(V\mathbf{x})}{\delta\mathbf{q}})\dot{\mathbf{q}}, \tag{2.3}$$

where $h'(\mathbf{x})$ is the resulting matrix from the derivate of $h(\mathbf{x})$. $\mathbf{q}$ has here been differentiated with respect to time and $\frac{\delta(V\mathbf{x})}{\delta\mathbf{q}}$ defines the derivative of the transformed point $V\mathbf{x}$ with respect to q with a $4 \times n$ matrix. Changing the notation for compactness can be done by defining the $2 \times n$ matrix

$$J = h'(V\mathbf{x})\frac{\delta(V\mathbf{x})}{\delta\mathbf{q}}, \tag{2.4}$$

such that

$$\dot{\mathbf{p}} = J\dot{\mathbf{q}}. \tag{2.5}$$

Although equation 2.5 shows a linear relationship can the matrix J not be solved unless it is square and of full rank, which it seldom will be. This is consistent with equation 2.1 since $h(x)$ is nonlinear and in general so is $V\mathbf{x}$. This prohibits an analytical solution to

the constraint. Instead the camera parameters are constrained such that the interaction follows the contact points with a minimized least-square error

$$E = \frac{(\dot{\mathbf{q}} - \dot{\mathbf{q}}_\mathbf{0})^2}{2} \text{ subject to } \dot{\mathbf{p}} - J\dot{\mathbf{q}} = 0, \tag{2.6}$$

where $\dot{\mathbf{p}} - J\dot{\mathbf{q}} = 0$ is the enforcing constraint that ensures the image point velocity to be consistent with the contact point's movement. Rather than choosing a minimum error, Gleicher et al. choose to limit the error E such that its gradient must be a linear combination of the gradients of the constraints.

$$\frac{dE}{d\dot{\mathbf{q}}} = \dot{\mathbf{q}} - \dot{\mathbf{q}}_\mathbf{0} = J^T\lambda \iff \dot{\mathbf{q}} = \dot{\mathbf{q}}_\mathbf{0} + J^T\lambda \tag{2.7}$$

Equation 2.7 is the combination of the two constraints in a matrix equation for some value of the two-vector λ of *Lagrange multipliers*. The only thing left to do is to update the camera parameters $\mathbf{q}$ with its derivative $\dot{\mathbf{q}}$ through various difference schemes such as *Euler's method*, central difference or higher order methods.

2.3 Using Screen-Space

Many research efforts have gone into the the formulations based on freely rotating, translating, and scaling in the field of touch interaction, such as the Fluid Integration of Rotation and Translation study by Kruger et al. [9] or the older 1992 study by Rubine et al. [10]. Previously rotation, translation and scaling were generally done sequentially which resulted in Screen-Space formulations to be an improvement in interaction efficiency.

A more recent publication called Screen-Space that was proposed by Reisman et al. [2] has been the focus of this thesis. Reisman et al. extended the control of 4 DOF, generally translation as well as one orientation axis, to all six degrees of freedom which in effect removed the need of any explicit gesture processing and took the screen-space principles from 2D to 3D. Much like equation 2.1, a function is defined which maps a point $\mathbf{x}$ in 3D model view, manipulated by the DOF parameters $\mathbf{q}$ and projected back to screen-space point $\mathbf{p}$.

$$\mathbf{p} = s(\mathbf{x}, \mathbf{q}) = h(PM(\mathbf{q})\mathbf{x}), \tag{2.8}$$

where h as before is the view matrix, P is the projection matrix and M is a matrix parameterized by the vector q which maps $\mathbf{x}$ into world-space. M is, as before, a combination of several rotation, translatation and scaling matrices. By keeping the projection matrix P constant, the only way to alter a model view point's corresponding point in

screen-space is by the local to world-space transformation, as no change is done to any of the camera's DOF.

Instead of using a difference scheme as Gleicher et al. [8], the best-fit transform parameters $M(\mathbf{q})$ is found by minimizing the non-linear L^2 error function that measures the total squared distance between the contact points' screen-space projection and their corresponding screen-space target positions. The error is defined as

$$E = \sum_i ||s(\mathbf{x_i}, \mathbf{q}) - \mathbf{p_i}||^2, \tag{2.9}$$

where $\mathbf{x_i}$ and $\mathbf{p_i}$ are the positions of the i^{th} object-space contact point and screen-space target points, respectively and the minimization of E is with respect to $\mathbf{q}$. Note that in cases with a large number of contact points the minimization method becomes oversaturated as it only has control over six DOF in three dimensions and thus have no perfect solution. Unless deploying an alternate scheme, the effect is seen as an even spread of slippage across all contact points.

The type of transformation, and particularly its DOF, is central in ensuring the desired interaction. As the screen-space formulation by Reisman et al. aims to extend previous RST methodology to 3D, the type of desired transformation $M(\mathbf{q})$ must contain rotation, translation and scaling matrices. However, the DOF which parameterize these transformations interact with the projection matrix P which can cause different parameters to generate the same results. For example, translation of an object towards the camera with an orthographic projection will not change the object size in screen space, while with a perspective projection it is indistinguishable to scaling. The seven DOF (translation, rotation and scaling) that a user could have control over may not be useful. The screen-space paper assumes the use of a projection matrix and thus ignores the redundant scale parameter to have control over a total of six DOF in 3D. The transform $M($**q** and its vector parameters **q** can then be defined as

$$\mathbf{q} = \begin{bmatrix} t_x & t_y & t_z & q_x & q_y & q_z \end{bmatrix} \tag{2.10}$$

$$M(\mathbf{q}) = T(t_x, t_y, t_z)Q(q_x, q_y, q_z), \tag{2.11}$$

where T is a translation matrix and Q a unit-quaternion matrix such that

$$q_w = \sqrt{1 - q_x^2 - q_y^2 - q_z^2}. \tag{2.12}$$

Multiple options exists for minimization methods of the non-linear least-square error, including the derivative scheme used by Gleicher et al. [8]. Reisman et al. chose to employ a Levenberg-Marquardt algorithm (LMA) [11] to ensure control to remain responsive without any high-cost in performance. LMA imposes a constraint in that it can only control as many DOF as the number of terms that gets sent into it. Each new contact point adds two terms to equation 2.9 as each image point contains 2D coordinates. This mean that with one contact point the formulation can only control two DOF, with two-point interactions four DOF and so on. Effectively, this binds an additional of sets of two DOF to the current number of contact points. This does not cause any issue when mimicing the standard RST-style synergy for both single and dual-point interactions. All of the transform's DOF are included for minimizations, involving three or more contact points. Reisman et al. found that their method had support of a number of unique manipulations previous methods did not with the price of some unintended behaviour in cases of rotational extrema due to rotational ambiguities or rotational exhuast.

Ambiguous rotations appear in normal use with three-point interactions. If two contact points remain stationary while the third is brought closer to those points the rotation will occur in the camera's XY-plane. However, when all three points are at roughly equal depths both counterclockwise and clockwise rotation with respect to the camera view are solutions that bring the moving contact point closer to the others in screen-space. Which one of the two that is chosen is dependant on the direction the solver determine to decent in. Reisman et al. suggests two different methods to mitigate the ambiguity by weighting one descent over the other, making the solver biased with initial values of the $\mathbf{q}$ vector or determining rotational direction dependant on pressure input. Pressure input puts an extra requirement on the installed hardware but adds a half dimension in which the user can "push" a contact point into the screen. Biasing the solver with initial values involves binding movement direction to specific rotation directions, for example moving left on the screen indicate a desired clockwise rotation around the Y-axis.

Rotational exhuast happens when the object started out as rotated and the movement is inverted from the rotational ambiguity case such that one contact points is brought away from the two other stationary contact points. Eventually the object will rotate such that it is parallel with the view plane which then causes the solver to produce a translation in the camera's Z-axis instead. This transition in interaction behaviour is desired and expected, however during the transition a new, perceptually perpendicular to previous axis, rotation is introduced. By designing a plane out of the three contact points and considering the movement of its normal can such unexpected effects be monitored. During the transition the plane's normal typically becomes more and more parallel with the camera

view direction until it reaches a maxima and starts to decline. The undesired behaviour is introduced upon this change and can thus be mitigated by designing a function which looks for the minima of the angle between the centroid of the contact point **c**'s normal and the camera direction vector at **c** given transform parameters **q** with respect to time.

2.4 DS3 and Separation of DOF

Since 3D imposes six unique DOF, input devices that generate less perceptual structure than that struggle to control all aspects at once. Screen-space proposes control of the six DOF in an integral way, while previously mentioned Sticky-tools by Hancock et al. [3] a separation between the degrees of freedom by allowing two contact points to function in an RST-style while an additional third contact point indirectly control the rotation of the object around the axis built by the two contact points that touch the virtual object.

Martinet et al. [4] investigated both these methods and found the sticky-tools separation to be difficult. Instead they implemented the depth-separated screen-space formulation with a clear seperation between the control of translation and the control of rotation. To support this the DS3 builds upon letting the screen-space method control the orientation and the Z-technique, a previous publication by Martinet et al. [12], control the position.

The Z-technique is designed to help the depth control in 3D positioning. Upon the first touch point's contact with the surface, a ray is sent from the camera location through the contact point position and into the scene to find the closest intersected object. Positioning is now constrained to the plane parallel to the camera plane passing through the object center. A second contact point's relative motion will then control the depth of the object's position by backward forward movements on the screen. Using a non-linear continuous function that maps the object displacement speed to a scale factor can the depth control be based on the velocity of the contact point. Repeated touch and releases can be used to position the object at any arbitrary depth position.

Figure 2.1 shows how with one direct contact point objects can be translated across the view plane. Translation in depth is performed indirectly through a second, indirect contact point. Moving the indirect contact point up along the screen results in the virtual object's depth increasing and downward movement moves it closer to the view plane. When two or more direct contact points are active, users can control only the orientation of the object through the same constrain solver as described in section 2.3. Figure 2.2 compares how the screen-space, sticky-tools and DS3 formulations manipulate different DOF under different types of interactions. Circles connected with a line represent the

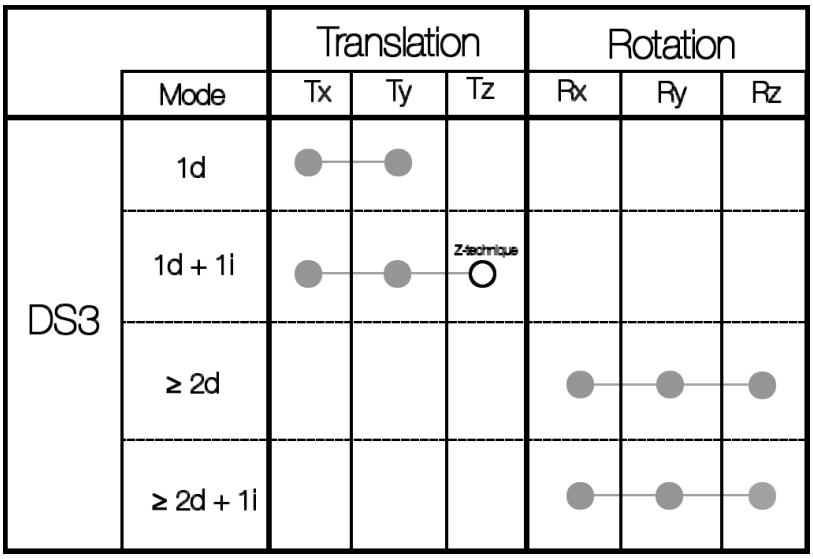


Figure 2.1: Formulation of Depth-Separated Screen-Space with the help of Z-technique. 1d stands for one active contact point which is in direct contact to the virtual object, 1d+1i is thus with an additional indirect contact point to allow control over an additional DOF.

DOF that are integrated under that type of interaction, empty circles represent the indirect control the additional indirect contact point gives.

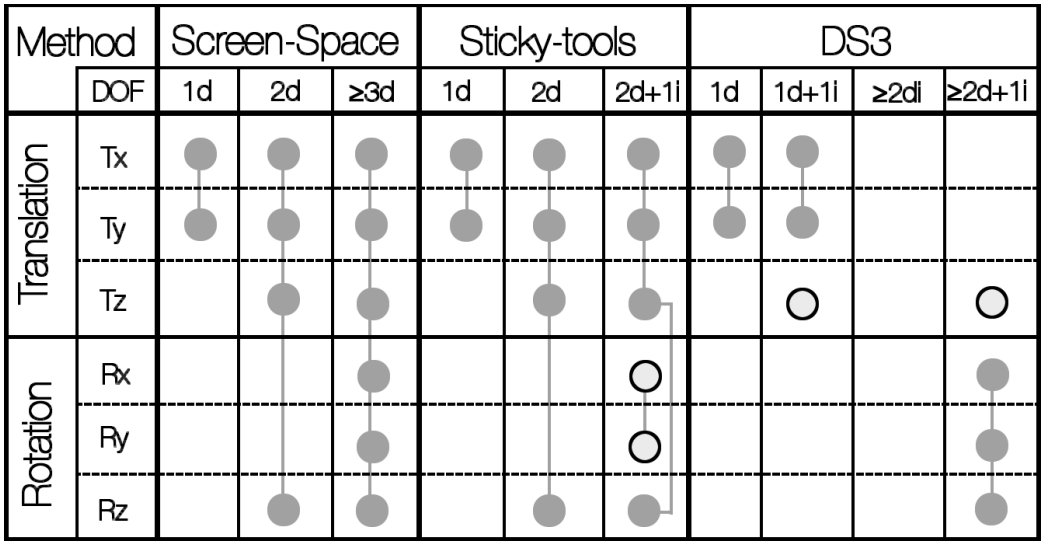


Figure 2.2: Comparison between the existing approaches' control of the 6DOF, based on the number of contact points and their directness (e.g., 2d+1i: two contact points directly touch the virtual object with one indirect finger).

A recent publication by Yu et al. [13] proposed a solution named Frame Interaction with 3D spaces (FI3D) to control seven degrees of freedom without requiring present objects to constrain the mapping. Such a claim is definitely of high interest in this thesis

use case. However, unlike other formulations it comes with a drastic cost of screen real estate as it requires an interface frame for its gesture interpretation. Due to the loss off screen area and enforced two hand interaction was the model not further considered. Another study, by Liu et al. [14], introduced a different method for 6DOF manipulation independant of directness to a present object in the scene using exclusively two-finger interactions. Their study proved the formulation to be efficient on small to medium touch screens. However, its usefullness on exponentially larger touch screens quickly decays as the interface was designed for use with a thumb and index finger and has thus not been used in the thesis.

3. Theory

This chapter goes through the mathematics behind OpenSpace's choice of DOF formulation, discusses the complications the formulation causes to an analytical gradient for the LM algorithm together with a tracing algorithm to find different celestial bodies.

3.1 Choices in DOF Separation

The choice to implement screen-space over other formulations for astronomical visualizations seem natural, as all other methods require an indirect contact point to perform all types of interactions. As such the controlled object are not allowed to outgrow the viewplane used for the manipulations, in which case contact points outside of the objects surface would become impossible. To allow control over a celestial body from a space perspective as well as a street-view perspective, the chosen direct-manipulation formulation had to be the screen-space method by Reisman et al. [2]

The methods mitigating ambiguous rotations and rotational exhuasts explained in section 2.3 have not been implemented in this thesis, as the chosen DOF setup heavily differs from the one discussed in the screen-space publication. Reisman et al. [2] arranged the control of an objects six degree of freedom, however in OpenSpace the objects, or celestial bodies, themselves cannot be moved or rotated. Instead instead the camera state, and thus its view plane, is manipulated during the user interaction. Due to the vastness of space and inexplicit direction of up or down, the desired control in an astronomical visualization software is vastly different from the user case of looking at one object. As such, all camera movements have been designed such that it is always done in related to a focus. A focus can be any type of celestial body, be it the sun, earth or a smaller asteroid. This naturally creates a new definition of the DOF. The screen-space formulation from equation 2.8 is set up such that matrix V is a combination of multiple transformations including all six DOF as

$$V = T(t_x, t_y, t_z)Q(q_w, q_x, q_y, q_z), \tag{3.1}$$

where each added contact point explicitly adds two DOF to the transformation. V is then multiplied to the current camera transform to get the final camera state C. In OpenSpace's case, the DOF are instead defined through:

- **Orbit** - contains two euler angles, θ and ϕ, to decide the camera's position and look-at direction related to the focused celestial body.
- **Zoom** - single value related to the distance to the focus.
- **Roll** - contains one angle value that determines the roll orientation of the camera.
- **Pan** - consists of two angles that determine the local look direction of the camera itself.

This formulation allows the user to freely traverse with the camera through space with control of all its DOF while keeping the celestial bodies in context. In order to achieve this, the camera orientation matrix Q needs to be separated in local and global rotations. This is generally done through a *look-at matrix* that takes the camera location, the direction it is facing as well as the up-direction and computes a resulting matrix L. L can then be used to separate Q by defining a global rotation $R_g = normalized(L^{-1})$ and a local rotation $R_l = R_g^{-1} \cdot Q$, such that

$$Q = R_g \cdot R_l. \tag{3.2}$$

Equation 3.3 is then a three-part way to describe the desired DOF separation. The first part calculates the rotation and translation transformations from the orbit DOF, second the zoom translation and third the local orientation of the camera, ie. roll and pan. The translation terms from the orbit and zoom are merged for notational compactness.

$$C = (\overbrace{(T_d R_g R_o R_g^{-1} + T_c)}^{Orbit\ translation\ =\ T_o} - \underbrace{\hat{T}_d \cdot z}_{Zoom}) \cdot \overbrace{\frac{Q(M^{-1})}{|Q(M^{-1})|}}^{Orbit\ rotation} \cdot \underbrace{R_l R_r R_p}_{Roll\ and\ Pan}, \tag{3.3}$$

in which $Q(M)$ is the quaternion of the 4×4 look-at matrix

$$M = \begin{bmatrix} \vec{s} & \vec{u} & -\vec{f} & \vec{0} \\ 1 & 1 & 1 & 1 \end{bmatrix}, \tag{3.4}$$

where $\hat{f} = -\hat{T}_o$, $\hat{s} = \hat{f} \cdot (R_g \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix})$ and $\vec{u} = \hat{s} \cdot \hat{f}$. R_g and R_l are defined as above, $T_d = T - T_f$ is the distance matrix between the camera position and the celestial

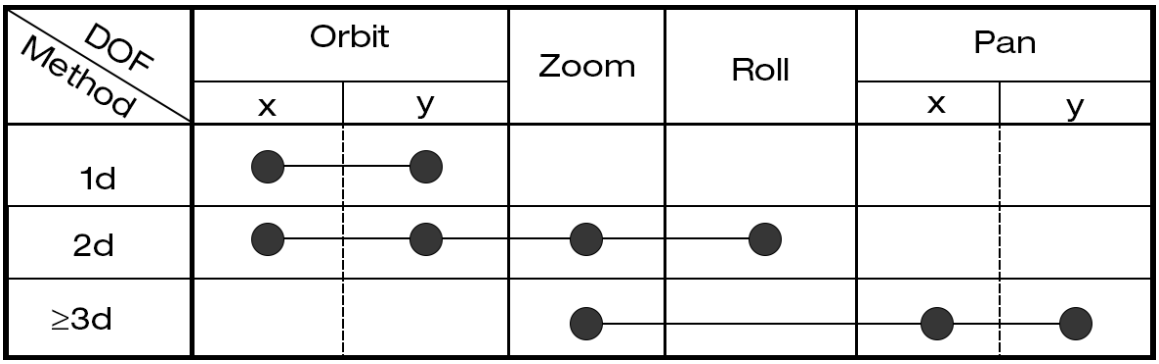


Figure 3.1: OpenSpace's DOF separation for direct-manipulation control. With the help of one, two or more active contact point that touches the virtual object.

body in focus. T_o is the translation part of the orbit-interaction while R_o is the rotation quaternion built from the orbit angles. Lastly R_r and R_p are quaternions gotten from the roll and pan angles respectively.

With equation 3.3 it is now clear how the camera state changes dependant on the newly defined DOF seperation. The resulting control implemented into OpenSpace can be seen in figure 3.1, where $1d$ represents one active contact point on the surface of a virtual object, $2d$ two, etc. The $\geq 3d$ case generally has control over all six parameters, however the orbit and roll DOF were muted by normalizing their individual partial derivatives. As to why this was done will be discussed in detail in section 5.1.

3.2 Creating the Minimize Function and its Gradient

The Levenberg-Marquardt algorithm is at the core of direct-manipulation interaction and is used as for a non-linear least-square curve fitting. As the algorithm interpolates between the Gauss-Newton algorithm (GNA) [15] and gradient descent, it naturally requires the gradient of the function that is subject to be minimized to operate.

Like discussed in section 2.3 the function $s(\mathbf{x}, \mathbf{q})$ maps the local point $\mathbf{x}$ to world coordinates to later be projected back to screen-space for location comparison with its respective contact point. The main difference from equation 2.8 however is that the transformation $M(\mathbf{q})$ parameterized by the DOF vector is not applied to the virtual object itself. The point $\mathbf{x}$ remains constant while instead the camera position and orientation, and thus the view plane to which $\mathbf{x}$ will be projected back to, is changed according to equation 3.3. As such, the minimization function becomes

$$E = \sum_i ||h(PC(\mathbf{q})T_f Q_f \mathbf{x_i}) - \mathbf{p_i}||^2, \tag{3.5}$$

where T_f and Q_f are translation and rotation transformation that maps point $\mathbf{x}$ from its local view to world coordinates. P is the projection matrix, $\mathbf{p_i}$ the corresponding screen-space point, $C(\mathbf{q})$ according to equation 3.3 and h the function that converts the 3D point to screen-space coordinates. E is minimized with respect to the vector $\mathbf{q}$ that is constructed by the parameters controlling the different DOF. Depending on the type of interaction $\mathbf{q}$ thus contains two, four or six elements.

The issue that arise is how the function's gradient changes dependant to $\mathbf{q}$. An analytical solution for this becomes complex as the statement is non-linear and contains transforms between 3D to 2D into a scalar value. Instead, a forward Euler method scheme is used such that

$$\nabla E_q = \frac{E_{q+h_q} - E_q}{h_q}, \tag{3.6}$$

where the step size h_q is a fraction of the respective value in $\mathbf{q}$. This means that the step size for the partial derivative changes as the DOF the function is derived with respect to changes.

Due to the large size differences in astronomical datasets, this raises a computer numerical limitation issue in the choice of step size h_q. Interactions under a street-view-like distance with a small asteroid or at distances approaching an astronomical unit (A.U) [16] should feel just as natural to the user. The extreme variance in interaction motion causes the parameters in $\mathbf{q}$ to fluctuate massively. For example, the zoom-parameter z in $\mathbf{q}$ determines the distance the camera should move towards or away from the focus. The identical interaction gesture on a large celestial body generates a greater traversed distance in depth compared to a smaller planet, resulting in vastly different z values. To keep the gradient ∇E_q convergent, the minimal step size h_q must be large enough such that the numerical difference

$$D = |E_{q+h_q} - E_q| > 0 \tag{3.7}$$

can be quantified.

It is also a topic of importance for optimization purposes as the number of iterations LMA requires to converge can drastically be reduced with an appropriately defined numerical gradient. A large step size makes the LMA more prone to overstep resulting in an increased number of iterations, while a step size small enough to generate a next to indistinguishable difference D, faces the opposite problem. The local minima is somewhere in between, however an extensive analysis for the best-fit step size h_q for each iteration is not feasible in real-time. Instead an iterative method were implemented to find the fitting step size in between the two local maximas. After defining the minimum viable step

size it is then scaled according to the size of the interacted object. Lastly the scaled step size is constrained to prohibit it from becoming larger than a $\frac{1}{1000}\mathbf{q}$ of the corresponding parameter in $\mathbf{q}$.

3.3 Finding Selected Objects

In order to perform any type of minimization is the local 3D point $\mathbf{x}$ required. This point is generated through general ray trace by sending a vector from the camera, through the contact point in screen-space and into the 3D scene.

$$\mathbf{v_r} = V \cdot (P^{-1} \cdot \begin{bmatrix} x \\ y \\ -1 \\ 1 \end{bmatrix}), \tag{3.8}$$

where V and P are the view and projection matrix respectively while x and y the contact point's coordinates on the image screen. All celestial bodies in OpenSpace have a rough bounding volume estimated by the objects radius, which allows for several methods to find if and where intersections occur between the line in the ray direction $\mathbf{v_r}$ and an object in the scene. If $\mathbf{v_o}$ is the vector from the camera position $\mathbf{c}$ pointing towards an objects position $\mathbf{p}$, then one way to define the distance between the line and the object is through orthogonal projection. Note that $\mathbf{v_r}$ is the direction of the traced line with the the camera position on it, and thus of unit length one. Another, computionally more efficient, method is to use the area of the parallelogram formed by vector $\mathbf{v_o}$ and $\mathbf{v_r}$. If the shortest interval between point $\mathbf{p}$ and the projected line is then less or equal to the bounding sphere's radius r, an intersection is considered.

$$|\mathbf{v_r} \times \mathbf{v_o}| \leq r. \tag{3.9}$$

Disregarding the case where $\mathbf{v_r}$ is the tangent to the bounding sphere's brink, there will always be two intersection points along the traced line while only the closest point with respect to the camera is needed. Equation 3.10 defines how far t along $\mathbf{v_r}$ starting from the camera position it is required to traverse in order to reach the closest intersection point.

$$t = \mathbf{v_r} \cdot \mathbf{v_o} - \sqrt{r^2 - |\mathbf{v_o}|^2 + |\mathbf{v_r} \cdot \mathbf{v_o}|^2} \tag{3.10}$$

Note that if the summation under the root equals to zero the traced line tangents the bounding sphere and only one intersection point exists. The desired local 3D point $\mathbf{x}$ can then be determined by

$$\mathbf{x} = Q_f^{-1} \cdot (\mathbf{c} + \mathbf{v_r}t - T_f), \tag{3.11}$$

where Q_f and T_f like before are the translation and rotation matrices that transform a point from its local view to world coordinates. This step is done to ensure that a surface position on an object remains static, since its' world coordinates will change with regards to time.

4. Implementation

This chapter discusses how the theory was implemented into OpenSpace. Starting with how the input is received from the touch surface to the scheme the multi-touch interface is built upon.

4.1 Decoupling the Input

The first stage in developing multi-touch interactability is to get the input. The touch protocol chosen for this was TUIO [7], which works through server-client transmission. The client was implemented into OpenSpace as a listener, receiving input through internet protocols in the form of *TuioCursor* class objects. The *TuioCursor* contains a range of sophisticated information about the input such as acceleration, pressure input or various position filters. However, the data OpenSpace uses can be reduced to

Listing 4.1: TuioCursor object

```
class TuioCursor : public TuioPoint
        double x;
        double y;
        double velX;
        double velY;
        TuioTime time; // stores time object has existed
        int id;
        std::list<TuioPoint> path;
```

Since multiple contact points can be used at once was there need of a list to separate the different touches. This is done through a $vector < TuioCursor >$ as seen in figure 1.2. When the TUIO server recognizes a change on its tangible surface a callback is sent to the client to update according to the change. In this thesis only three kinds of callbacks were used.

- **addCallback** - adds a new *TuioCursor* to the list of contact points.
- **removeCallback** - remove the entry from the list with respective *TuioCursor* id.

- **updateCallback** - push back the old *TuioCursor* position and time as an entry in the *list<TuioPoint> path* and update the object's state itself.

The add-callback is called once a new contact point touches the tangible surface, remove once a contact point is removed and update if a current contact point changes its state. Through the update-callback could a tap-tracker be introduced by looking at the time a contact point had existed and its total movement over the screen.

Since the server is a separate process from OpenSpace, it is imperative to decouple the receival of input from the programs frame rate. Otherwise the input would vary when running the program on a low-end system from a high-end one. This was done by creating a list in the *TuioEar* object that only changes under mutex locks and having the module class copy the list when the program started to render a new frame. The last thing to consider was a safeguard against the case in where OpenSpace would render faster than the client was receiving input. If this were the case, no new input would have been received from the last frame and the desired interaction already executed. By forcing interaction to only be done with new input, two processes, OpenSpace and the TUIO client, could run concurrently while producing identical interaction behaviour given the same input on different machines.

4.2 Putting Markers in Screen-Space

In order to improve the presentation capabilities of the multi-touch interface it was important to render what type of interaction was performed on the touch screen. Customizable circle markers were implemented for this with control over size, color, opacity o as well as thickness. Figure 4.1 shows how the thickness parameter t manipulates the fall-off function that defines the level of opacity at distance $d \in [0, size]$ from the contact point coordinates. Assuming within the size radius, the alpha level a for a pixel can thus be defined by

$$a = min(d^t, o). \tag{4.1}$$

4.3 Determine Type of Interaction Scheme

In regards with what to do with the input, it must first be settled which interaction-scheme is to be used. In conjunction with the screen-space direct-manipulation formulation a velocity-based supplementary method have been added to handle the cases screen-space solver does not.

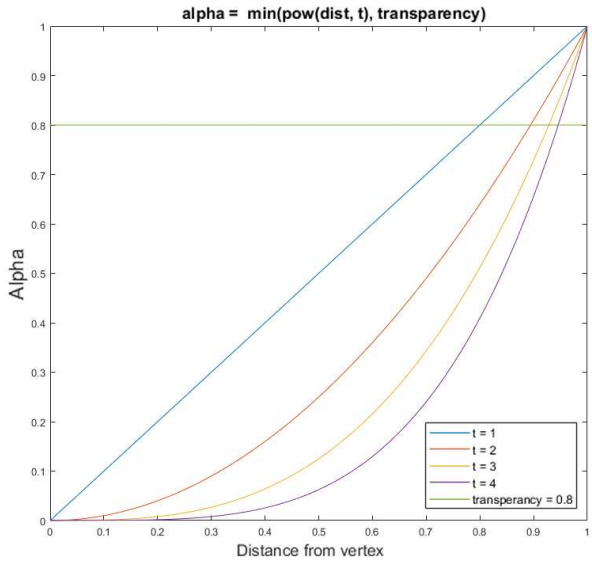


Figure 4.1: Alpha fall-off function dependant on the choice of thickness and transperancy cap

The velocity-based method first interprets the type of interaction to then add a velocity to the desired DOF to then decelerate with time to a chosen friction value. Exactly how this is done will be explained below in section 4.4 and 4.5. As these two methods are distinctly different, one maps the exact interactions through constraints and the other maps input movement to an arbitrary velocity in a specific DOF, it is imperative for them to share a similar formulation to minimize confusion. This is achieved by having both share identical DOF formulations for control and analogous gestures. Sharing identical DOF scheme is no issue; it even simplifies the development as the updated camera state computation can be shared by simply choosing where it receives the parameters which are to be used for the update. With analogous gestures the velocity-based method is constrained to use equivalent gestures for specific DOF control as figure 3.1.

Another important aspect in intertwining the two schemes is to automatically and seamlessly swap between them. Direct-manipulation can be used in all cases where every active contact point touches a virtual object but fall short when generally traversing through space. This since during those interactions there is no contact with celestial bodies and thus no local 3D point $\mathbf{x}$ can exist. Velocity-interaction struggles with scaling values deciding the magnitude of the velocity when the context changes drastically. Getting appropriate scaling values when the context can vary from a detailed street-view on an object to traversing through entire solar systems is not a trivial task. A decision solver was implemented to elect the type interaction scheme based on two queries:

- Are all contact points on the surface of a virtual object?
- Is the focused object large enough on the screen?

If both are true direct-manipulation can be used while the velocity-method takes all other cases. The first condition can be determined through the methods in section 3.3 but the second depend upon a size definition of an object in image space. Since the object's radius r in the world is known and the projection matrix P scales objects depending on the distance to the camera can the scaling coefficients be found by choosing an arbitrary vector $\mathbf{v}$ as well as an abitrarily choosen distance d. The scaling value s can be defined as

$$s = \frac{|P \cdot \mathbf{v}|}{|P \cdot (\mathbf{v} + \begin{bmatrix} 0 \\ 0 \\ d \end{bmatrix})|}, \tag{4.2}$$

where $\mathbf{p}$ and $\mathbf{c}$ are the positions of the object and camera respectively. s defines the amount the radius of an object decreases as the distance from the camera increases. Note that it makes no difference to what axis d is applied to. An object-specific screen radius r_s can thus be defined as

$$r_s = \frac{r}{(|\mathbf{p} - \mathbf{c}| - r) \cdot s}, \tag{4.3}$$

where r is the original radius of the celestial body. r is used in the denominator to measure the distance between the camera and an object's surface rather than its center. $r_s \in [0, r]$ defines the star, planet or asteroid's size on the image screen. As the view plane is defined in normalized device coordinates (NDC) before aspect ratio distortions, an object with $r_s = 0.5$ would cover the entire y-plane on parts of the screen. The threshold h_r which $r_s \geq h_r$ to activate direct-manipulation can be chosen readily by the user, however $h_r = 0.2$ gave the best results from user tests during development.

4.4 Interaction Types and Interpretation

In order to apply a velocity to the desired DOF the input has to be interpreted and classified to a distinct interaction. There are five different kinds of gesture interpretations:

- **Refocus** on a celestial body can be done by double tapping, a double tap in space decreases the ditance between the camera and the focused object by a fixed about.
- **Orbit** control around the focus is done by single finger interaction.

- The **zoom** DOF is controlled by two or more contact point with the preponderance motion of increasing the distance between a contact point and their centroid.
- The **roll** value determined by the total angle change between contact points and their centroid or a static contact point.
- **Panning** is done by having three active contact points which average distance between each other remains constant.

As the orbit control is the only gesture with one contact point are there no conflicts in the interpretation. The choice to pan with three, in relation to each other static, contact points is to maintain the similarity with the direct-manipulation control. Although the screen-space formulation give access to more than just pan DOF it is not distinctly different and continue to interact in a way users expect. The ability to traverse through space is greatly improved by allowing the user to refocus on a new celestial body through double taps.

However, pan and zoom are only separated by the type of movement and can have overlap in its interpretation. For example, movement with two contact points solely along the y-axis or x-axis generate a huge changes in angle to the centroid in a way that is not expected. A way to limit this behaviour could be to lock the roll to be the angle change related to one static contact point instead of the centroid. User tests showed that this was a natural gesture for controlling the roll when interacting with a small area of the touch surface but decreased once larger screens were in use. Instead a safeguard for this specific interaction was implemented, allowing both gesture versions to generate DOF control over the roll.

4.5 Velocity-based Method

After interpreting the type of gesture, the motion of the input are used as the main variable determining the magnitude of the added velocity in the determined DOF direction. Each DOF requires a sensitivity scalars in order to ensure consistency with different screen sizes and aspect ratios. The zoom DOF is particularly interesting as its expected velocity value largely depends on the context the pinch-gesture is performed with. Keeping the sensitivity static would either make interaction illsupported when traversing from one celestial body to another or when the camera comes closer to an object. Using the distance between an objects surface and the camera position as a linear scale factor resolves cases where the distance is large to medium, however the issue remains on distances comparable the radius of the sun.

As the interaction adds velocity, to prevent infinite movement this has to decay with time. DOF specific friction parameters were introduced to control this deceleration. The computation can only be done once for each frame, which caused inconsistency in the velocity deceleration between different systems or inconsistent render times. The deceleration was thus decoupled from the frame time by introducing a time tracker. Each frame, the time since last computation is considered to determine how much the velocity should decelerate such as

$$\mathbf{v}' = \mathbf{v} \cdot (1 - f)^{\lambda}, \tag{4.4}$$

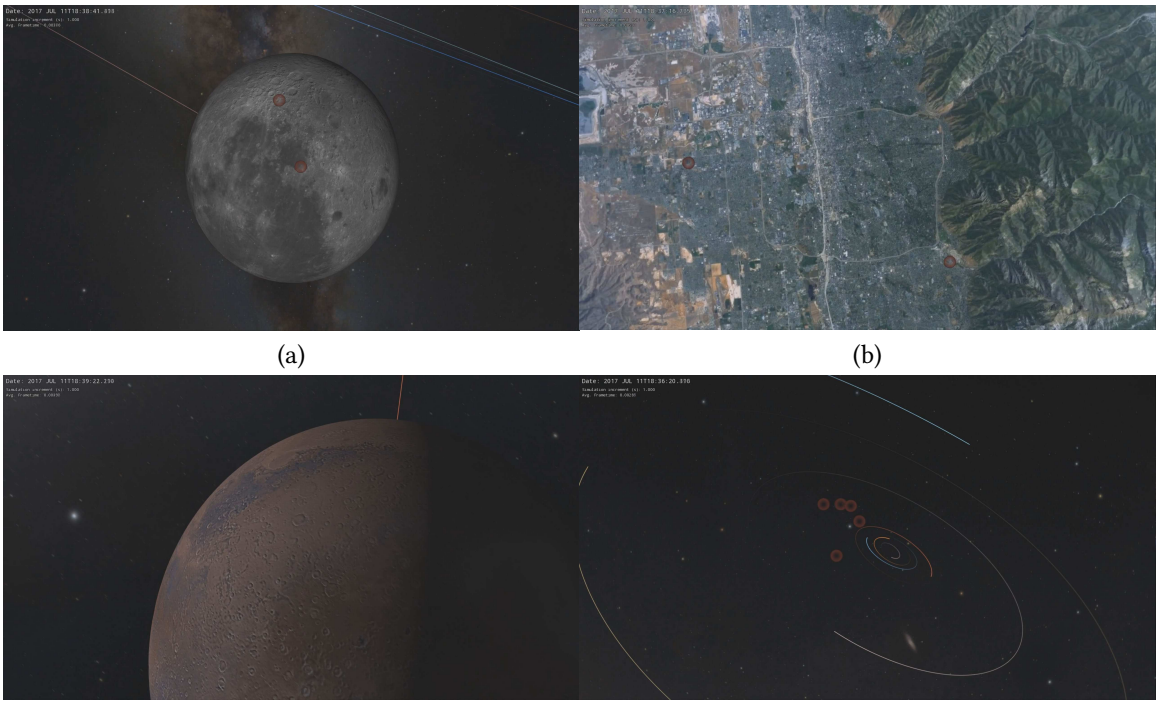
where f is the friction scalar and λ the strength of decay controlled by

$$\lambda = \frac{dt}{h_f}, \tag{4.5}$$

with dt as the frame time and h_f as the decoupled decay frequency.

5. Results

This chapter brings up the resulting multi-touch interface and discusses why the implemented methods were used over others alternate approaches. With the described implementation the user can interact with any celestial body in the scene and traverse in an expected way through the multi-touch gestures. Figure 5.1 shows the resulting navigation interface in use in both close range over the Salt Lake City area in Utah, USA on 5.1b and when traversing the solar system in 5.1d.



(a) (b)

(c) (d)

Figure 5.1: Shows different interactions in OpenSpace. (a) and (b) has two contact point interacting with the Moon and Earth respectively to control the camera. (d) illustrates the gesture to control the roll DOF.

The gradient in the levenberg-marquardt algorithm had to be computed numerically through finite difference rather than analytically, which resulted in discussed issues relating to the used step size. Central difference as well as higher order schemes were

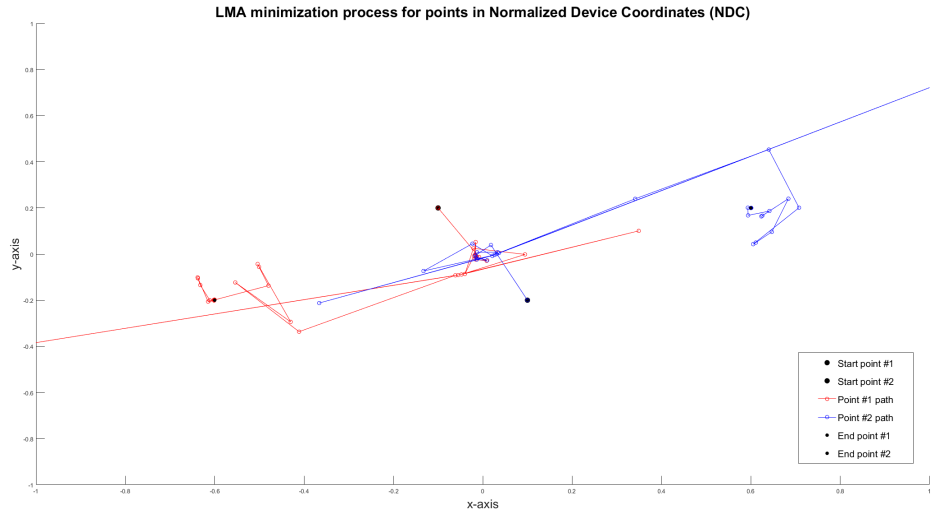


Figure 5.2: The red and blue lines are the LMA minimization paths for each iteration step for respective contact point for this particular interaction. Large circles denotes the contact points initial location and the smaller ones the goal position.

compared with the implemented forward difference. Although higher order schemes offered a higher accuracy on the resulting gradient it became a performance issue since the calculations had to be done for each contact point as well as LMA iteration, which could vary between 30 to 200 before converging. The forward difference generated the highest number of required iterations but still maintained the most effective method in terms of performance, especially in cases with more than one contact point. Figure 5.2 shows the path the algorithm took in screen-space with two contact points for each iteration towards its goal position. Note that as the least-square error is non-linear to the camera parameters $\mathbf{q}$, which results in the seemingly erratic iteration steps.

5.1 Discussion

Evaluating a user interface is a subjective matter. An interface is intuitive only when users understand its behaviour and effect without use the of experimentation, reason or assistance. Such intuition requires prior knowledge, either from experience in the real world or with other software. One way to assist the interface's innateness is to match the number of controllable DOF with the with the structure of the input data. For example, using a 3D mouse or a handheld six DOF tracking device allows for intuitive direct manipulation in as many dimensions concurrently. This was the main reason why the ≥ 3 case for direct-manipulation was limited to the control of only three DOF. A tangible surface can only generate 2D, or 2.5D if the surface is pressure sensitive, input points on multiple

locations on the screen simultaneously. This allows for clear control in up to four DOF, however not more. There was also next to no drawback in this decision as all six DOF remained controllable, albeit under different gestures.

Several different ideas for DOF formulations were considered during the thesis, particularly ones without a relative focus point in the scene. Some users expressed initial confusion that the zoom parameter took them closer or further away from an object and not in the direction the camera was looking. One way to alleviate this would be to keep the camera free of any relations and movement would be done according to its orientation. However, this raises a more concerning regarding finding new points of interests from far distances. Even experienced users would struggle to find their way between planets in the solar system, not to mention a user at public exhibitions. As exhibition use were the focus for this thesis it was determined that focus points in the scene were necessary in order to maintain celestial bodies in context over empty space.

The velocity-based method initially had control over the zoom and roll parameters with two or more contact points, much like the direct-manipulation scheme. This was commented on during user tests as undesired. The depth control was far more common compared to the roll, which made the feature result in undesired *wiggle* back and forth of the roll during the pinch-to-zoom gesture motion.

Tests to how the screen size affected the gestures were done using a 55” touch table as well as a 5.5” smartphone. Scaling the sensitivities scalars of the velocity-based method became an interesting point of discussion to maintain consistency between real-world distances and resulting interaction in the software. It became apparent that a linear solution for this would not be desired as the way a tangible surface is handled by the user drastically changes with its size. For example, more than three contact points were never used on the smartphone device while it was a common occurance on the 55” screen. Users expressed they expected the smaller touch screen to generate an interaction with higher sensitivity much like the way it naturally did.

6. Conclusions and Future Work

This chapter answers the research questions and discusses potential improvements or extensions that could be made to the navigational interface for the future.

6.1 Conclusions

It has been shown that the combination of a velocity-based interaction model and a screen-space direct-manipulation scheme allows for a precise control of objects and effective traversal ability through the vastness of space. The user confusion when swapping between these interaction mode is minimized by having both methods share an identical DOF formulations and a similar set of interaction gestures. Since both schemes have the same formulation the computations to update the camera state can be shared and the parameters it would use are readily taken from the appropriate interaction mode.

To maintain the interaction intuitive in large 3D visualizations it is required to limit the number of controllable DOF for different gestures such that it matches the structure of the input device. By keeping all interaction in relation to a user chosen celestial body can the points of interest of the visualization be kept in context throughout the vast space. In order to safeguard the application from inconsistencies on different hardware systems it was required to ensure that all input that have happened since last frame is stored before use. With the velocity-based interaction mode it is also important to make sure that the decelerate phase is computed with respect to the time passed since last frame.

The evaluation on the screen-space direct-manipulation method is a non-linear minimization problem, as thus the Levenberg-Marquardt algorithm to solve the least-square error is a fitting solution to maintain the software performance. To resolve the numerical computation limitations for astronomical distances the quality of the numerical gradient LMA uses is maintained through a low-cost iterative method to find an appropriate step size h.

6.2 Future Work

The current multi-touch interface is sufficiently good for public exhibitions and general control of camera through the solar system. For increased control over the application the additions of buttons and sliders would greatly help. For example, dedicated buttons for specific camera states would help users to easily move back to a point of interest from their current perspective or different visualization layers to display data could help improve the understanding of space. Sliders could give the user direct control over the time dimension as well.

The linear distance scaling of the zoom parameter in velocity-based interactions could be improved to more accurately mimic the desired interaction by performing a more detailed analysis of input data in consideration of the context. This goes hand in hand with the interpretation solver and gesture conflicts between roll and zoom. Using machine learning could help the solver learn in which contexts different interactions are desired and favour them accordingly.

The direct-manipulation mode uses a rough bounding sphere to compute the local surface position on a celestial body. As planets and asteroids are seldom perfect circles, performing another type of intersection algorithm, such as Möller-Trumbore[17], to find the touched face of the object could help improve precise control of small arbitrarily shaped objects.

References

[1] Alexander Bock, Emil Axelsson, Karl Bladin, Jonathas Costa, Gene Payne, Matthew Territo, Joakim Kilby, Masha Kuznetsova, Carter Emmart, and Anders Ynnerman. OpenSpace: An open-source astrovisualization framework. *The Journal of Open Source Software*, 2(15), Jul 2017.

[2] Jason L Reisman, Philip L Davidson, and Jefferson Y Han. A screen-space formulation for 2D and 3D direct manipulation. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 69–78, 2009.

[3] Mark Hancock, Thomas Ten Cate, and Sheelagh Carpendale. Sticky tools: full 6DOF force-based interaction for multi-touch tables. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 133–140, 2009.

[4] Anthony Martinet, Gery Casiez, and Laurent Grisoni. Integrality and separability of multitouch interaction techniques in 3D manipulation tasks. *IEEE transactions on visualization and computer graphics*, 18(3):369–380, 2012.

[5] Robert JK Jacob, Linda E Sibert, Daniel C McFarlane, and M Preston Mullen Jr. Integrality and separability of input devices. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 1(1):3–26, 1994.

[6] Ken Schwaber and Mike Beedle. *Agile software development with Scrum*, volume 1. Prentice Hall Upper Saddle River, 2002.

[7] Martin Kaltenbrunner, Till Bovermann, Ross Bencina, and Enrico Costanza. TUIO: A protocol for table-top tangible user interfaces. In *Workshop on Gesture in Human-Computer Interaction and Simulation*, pages 1–5, 2005.

[8] Michael Gleicher and Andrew Witkin. Through-the-lens camera control. In *ACM SIGGRAPH Computer Graphics*, volume 26, pages 331–340, 1992.

[9] Russell Kruger, Sheelagh Carpendale, Stacey D Scott, and Anthony Tang. Fluid integration of rotation and translation. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 601–610, 2005.

[10] Dean Rubine. Combining gestures and direct manipulation. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 659–660. ACM, 1992.

[11] Jorge J Moré. The Levenberg-Marquardt algorithm: implementation and theory. In *Numerical analysis*, pages 105–116. Springer, 1978.

[12] Anthony Martinet, Gery Casiez, and Laurent Grisoni. The design and evaluation of 3d positioning techniques for multi-touch displays. In *3D User Interfaces (3DUI), 2010 IEEE Symposium on*, pages 115–118, 2010.

[13] Lingyun Yu, Pjotr Svetachov, Petra Isenberg, Maarten H Everts, and Tobias Isenberg. FI3D: Direct-touch interaction for the exploration of 3D scientific visualization spaces. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1613–1622, 2010.

[14] Jingbo Liu, Oscar Kin-Chung Au, Hongbo Fu, and Chiew-Lan Tai. Two-Finger Gestures for 6DOF Manipulation of 3D Objects. In *Computer Graphics Forum*, volume 31, pages 2047–2055. Wiley Online Library, 2012.

[15] MR Osborne. Separable least squares, variable projection, and the Gauss-Newton algorithm. *Electronic Transactions on Numerical Analysis*, 28(2):1–15, 2007.

[16] T-Y Huang, C-H Han, Z-H Yi, and B-X Xu. What is the astronomical unit of length? *Astronomy and Astrophysics*, 298:629, 1995.

[17] Tomas Möller and Ben Trumbore. Fast, minimum storage ray/triangle intersection. In *ACM SIGGRAPH 2005 Courses*, page 7, 2005.