8-8-2017

Exploring GRIA2 Sequence Variations Using Virtual Reality

Jimmy Fan Zhang
jfz8009@rit.edu

Exploring *GRIA2* Sequence Variations Using Virtual Reality

by Jimmy Fan Zhang

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Bioinformatics

Thomas H. Gosnell School of Life Sciences

College of Science

Rochester Institute of Technology

Rochester, NY

August 8, 2017

Rochester Institute of Technology
Thomas H. Gosnell School of Life Sciences
Bioinformatics Program

To: Head, Thomas H. Gosnell School of Life Sciences

The undersigned state that Jimmy Zhang, a candidate for the Master of Science degree in Bioinformatics, has submitted his thesis and has satisfactorily defended it.

This completes the requirements for the Master of Science degree in Bioinformatics at Rochester Institute of Technology.

Thesis committee members:

Name	**Date**
Feng Cui, Ph.D. Thesis Advisor	
Gary R. Skuse, Ph.D.	
Paul A. Craig, Ph.D.	
Joe Geigel, Ph.D.	

Michael V. Osier, Ph.D.
Director of Bioinformatics MS Program

475-4392 (voice)
mvoscl@rit.edu

Table of Contents

Abstract

The effective visualization and presentation of biological data is of critical importance to research scientists. The increasing rate at which experiments generate data has only exacerbated the problem. While bioinformatics datasets continue to increase in size and complexity, the shift to adopt new user interface (UI) paradigms has historically lagged. Consequently, a major bottleneck for analysis of next-generation sequencing data is the continued use of UIs primarily inspired from the 1990's through the early 2000's. This paper presents the novel use of virtual reality (VR) as a medium for visualizing genomic, transcriptomic and proteomic data. Using the *Gria2* (GluR2 or GluA2) gene and its associated gene products as our main objects of interest, we present Gria2-Viewer, a proof of concept software tool for visualizing any gene variant within the Gria2 locus. For any given genomic or transcriptomic variant of *Gria2*, we can quickly visualize its position on the protein subunit, rendered as a secondary structure. We also present a design for an experimental case study which compares our software versus a "traditional" workstation for ascertaining the severity of any Gria2 variant and its location within a 3d representation of the protein.

Introduction

During the mid to late 1990's, the medical research industry was focused on a complete mapping of the human genome. It was understood that new disease models depended on the accumulation of relevant genomic data, which was relatively sparse at that time. From the 2010's to the present day, the accumulation of vast quantities of biomedical data across various -omics fields present an altogether different problem: how can researchers effectively mine vast quantities of data to describe only the phenomenon for which they are interested?

In this paper, we explore the extent to which effective visualizations and user interfaces in VR may help researchers quickly identify important genomic variants identified in exome sequencing data. As part of this use case, we present a VR approach to visualizing sequence variations in *Gria2* identified through exome sequencing. In so doing, we hope to establish that VR software may be a valid alternative to traditional web-based tools currently available for genomics, transcriptomics, and proteomics, research.

Methods and Materials

Hardware Requirements

We have regular access to a VR-capable computer with the following specifications: Intel Core i7-6700HQ Quad Core processor, a 6GB GDDR5 NVIDIA GeForce GTX 1060 graphics card, and 8GB of RAM. We used Oculus Rift with the following specifications: Oculus App Version 1.16.0.409144 (409268), device firmware version 708/34a904e8da. To detect a user's hands, we used Leap Motion Software Version: 3.2.0+45899, Leap Motion Controller ID: LP22965185382, Firmware 1.7.0.

Development Tools

Our VR-enabled desktop application was built using Unity3D (Unity), a game development environment with virtual reality capabilities. It utilizes the C# programming language, which comes as part of Microsoft's .Net software. As one of only two serious game development platforms that are commercially available for free and support VR, there was a limited choice in terms of our development tools. The alternative platform is Unreal Engine. It is not widely used within RIT, and its active user base worldwide pales in comparison to Unity's [1]. To avoid any potential problem for which there might be limited campus and online resources, Unity was established as the development platform for our VR application.

A software package called UnityMol was created in Unity by Marc Baaden of Centre National de la Recherche Scientifique (CNRS). A non-VR 2014 version of the software is available for free download via SourceForge.net (Sweet UnityMol r676 beta r7) [2]. It is governed by a French copyright license called CeCILL-C which grants us the right to its free use and modification. Using Unity version 5.4.2f, Sublime Text 3 Build 3126 (sublime), a code editor, and Git 2.11.0.windows.3 (git) for version control, we were able to use the 2014 version (base code) as the basis of our VR application. While Baaden currently leads a software team which also focuses on bringing VR to UnityMol, we were unable to compile the source code from their latest iterations. In contrast, the base code compiled and proved amendable to our changes.

Gria2 Data

Source files containing structural information on the Gria2 protein was downloaded from Protein Data Bank (PDB) [3]. Prior studies have previously identified glutamate binding and closing mechanisms using PDB: 1FTO [4] with the caveat that 1FTO only captures the ligand binding core of the protein. A major contributor of *Gria2* protein information to PDB is Maria V. Yelshanskaya et al. In her 2016 publication, a novel allosteric binding site of *Gria2* is inferred from diffraction-quality crystallization of a modified Rattus norvegicus GluA2 AMPA receptor subunit [5]. Preliminary data provided by the Paciorkowski Lab suggests that *Gria2* missense mutations at the novel allosteric region characterized by Yelshanskaya et al may negatively affect brain development in children, leading to infantile onset epilepsy among other neurological disorders (unpublished) [6]. Of the five structures submitted to PDB through Yelshanskaya et al, 5L1B shows the Gria2 structure in apo state. 5L1B was selected for use in this project due to its symmetry and unbound nature.

Gria2 homologs were obtained for Mus musculus, Homo sapiens, Macaca mulatta, Pan troglodytes, and Gallus gallus from NCBI (Figure 1) with the purpose of deducing the conservation and hence relative importance of each nucleotide position [7].

Species	Common Name	DNA key	mRNA key	Protein Sequence key
Rattus norvegicus	Rat	>NC_005101.4:c179704629-179584302	>NM_017261.2:431-3082	>NP_058957.1
Mus musculus	Mouse	>NC_000069.6:c80803204-80682904	>NM_001083806.1	>NP_001077275.1
Gallus gallus	Red junglefowl (chicken)	>NC_006091.4:21570503-21667322	>NM_001001775.3	>NP_001001775.2
Macaca mulatta	Rhesus Monkey	>NC_027897.1:157221519-157371648	>NM_001185013.2	>NP_001171942.2
Pan troglodytes	Chimpanzee	>NC_006471.4:161117506-161261776	>NM_001184994.3	>NP_001171923.2

Homo sapiens	Human	>NC_000004.12:157205099-157370583	>NM_000826.3:460-3111	>NP_000817.2

Table 1: FASTA files were obtained from NCBI and downloaded into the StreamingAssets folder of the project. A parsing routine in ModelFASTA.cs is called as needed. It extracts from the header line of the relevant FASTA file the key using regular expressions.

The documents listed are preloaded into the StreamingAssets folder of the G2V project. G2V software builds have access to a compressed version of the StreamingAssets folder at runtime. During the User Interface Case Study, we plan to provide the equivalent of "built-in" access to these documents for subjects randomized to the Traditional Computer group. A chrome browser will be open with tabs corresponding to the NCBI page for these documents.

Software Design and Development

Basic Unity Objects. Software development within Unity constitutes its own specialty. A detailed discussion of Unity-specific challenges and best practices are outside the scope of this paper. However, basic Unity concepts are described in this section.

By virtue of being a game development platform, all objects in Unity are of type GameObject. A GameObject instance may contain zero or more components, such as a MeshFilter. A MeshFilter object contains a reference to a Mesh instance. A Mesh instance can represent a single polyhedron by virtue of its internal data structures: vertices, triangles, normals, and uvs arranged in arrays of the appropriate type. However, a single Mesh instance does have an upper vertex limit around 6000. In Gria2-Viewer, the primary objects of interest are GameObject instances which contain a Mesh instance [8]. A mesh can be accessed via the following: Mesh m = gameObject.getComponent<MeshFilter>().mesh assuming there exists a non-null reference to GameObject named gameObject.

Class	Description (source: Unity Docs)	Contains	Parent in Scene Hierarchy	Parent in Object Oriented Inheritance
GameObject	"Base class for all entities in Unity scenes."	Empty, Transform Component, other Components, or other GameObjects.	Other GameObjects, Unity scene	Object
Component	"Base class for everything attached to GameObjects. Note that your code will never directly create a Component. Instead, you write script code, and attach the script to a GameObject."	A reference to the GameObject to which the component is attached. A reference to the Transform attached to the GameObject to which the component is attached.	Is attached to a GameObject but has no parent since it isn't represented in scene hierarchy.	Object
MeshFilter	Accessor class for Mesh objects. Passes Mesh info to MeshRenderer for rendering.	Mesh object.	Is attached to a GameObject, but has no parent for the same reason as Component.	Component
MeshRenderer	"The Mesh Renderer takes the geometry from the Mesh Filter and renders it at the position defined by the object's Transform component."	References to Lighting, Material objects used to render geometry.	Is attached to a GameObject, but has no parent for the same reason as above.	Renderer > Component
Mesh	Represents a geometry, e.g. a plane, cube, or any other polyhedron.	int[] triangle, Vector3[] vertices, Vector3[] normals, Vector2 uv, Color[] colors	Retreived via MeshFilter.	Object
MonoBehaviour	Base class from which all Unity scripts derive.	Awake(), Start(), Update() functions	Scripts which inherit MonoBehaviour are attached to GameObject instances.	Behaviour > Component

Table 2: Common objects in Unity and their uses. Instances of MeshFilter, MeshRenderer, and other classes are often refered to as a type of Component when attached to a GameObject. The Scene Hierarchy refers to a directed acyclic graph (DAG) whose nodes represent GameObjects within the entire scene. Parent-child relations in the Scene Hierarchy refers to the relative grouping of objects in the DAG, whereas Parent in the Object-Oriented (OO) sense refers to inheritance.

Game Loop. The game loop is a ubiquitous concept in the gaming industry and influences Gria2-Viewer in subtle but important ways. A game loop is a finite state machine that describes the high level game state at any given time point. It is modeled roughly as follows: when the player enters the game for the first time, the game loop starts. The current game state is rendered

and displayed to the player. The player assesses the current game state and decides the next move, pressing the corresponding input(s). The next game state is computed based on the player's inputs and scene information. As soon as the next game state is rendered on screen, it becomes the current game state. The player assesses the newly created game state and responds with inputs, repeating the loop [9]. Unless game-ending conditions are triggered, the game loop continues.

Unity provides the MonoBehaviour class to help developers implement the game loop concept. The purpose of MonoBehaviour is to contain base code which organizes component actions into one of several game loop states. MonoBehaviour therefore contains the Awake(), Start(), and Update() functions, as well as other functions specific to Unity's implementation of game loop design.

To maintain game loop design principles, Unity encourages scripts to inherit from MonoBehaviour, but will otherwise compile and run normal C# classes with the caveat that those classes do not have the opportunity to directly affect game loop behavior [8].

VR applications rely on game loop architecture due to similarities in their state changes following user input. On start, the application takes on the state S_{START} and information is rendered into the headset. Information about the player's head orientation from the headset and finger positions from Leap Motion are gathered. The next state S_1 is then computed based on said information and rendered into the headset. Upon render completion, S_1 becomes the current game state, and input from the headset and Leap Motion determines the next state S_2. This continues to S_N so long as the user does not exit the application. The set of all states from S_1 to S_N is part of the larger "playing" game state within the game loop. In Gria2-Viewer, GameObjects that are subject to user input have some influence on the exact parameters of the next state and

are therefore part of the game loop (Figure 1). These GameObjects have attached components that inherit from MonoBehaviour and define the Awake(), Start(), and Update() functions as appropriate.

MVC. Changes and additions to base code relied heavily on the model-view-controller (MVC) design paradigm. MVC design calls for a separation of concerns into three different components of the software. The Model component is responsible for representing the data the software needs to interact with. The View component is primarily responsible for rendering and maintaining the correct views given inputs. The Controller mediates exchanges between Model and View components while also listening to user input events.

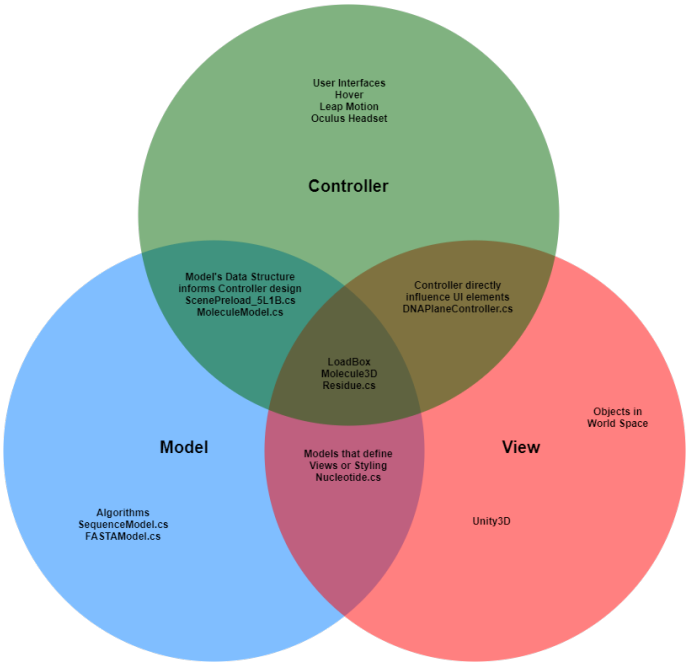


Figure 1: In building Gria2-Viewer, we attempt to faithfully execute on the MVC architecture, creating separate namespaces: VRModel, View, and Controller. Nevertheless some components within Gria2-Viewer blur the line between the different components of the ideal MVC model.

The separation of concerns in MVC architecture is meant to keep application logic apart from their presentation. In base code, modules primarily concerned molecular structure display reside primarily within Molecule.View, but can also be found in Molecule.SecondaryStructures or

within the surface folder without assignment to any particular namespace. Consequently, base code MVC architecture resembles a Venn diagram rather than cleanly separated MVC components (Supplemental Figure 3). During subsequent software development from base code onwards, we did not attempt any code refactoring aimed at separating MVC components into their respective namespaces. Rather, separate namespaces were created as folders within "/Script/" apart from base code. Modules such as Residue.cs, Nucleotide.cs, DNAPlaneController.cs, were added to their respective namespaces. Nevertheless, the separation of concerns problem remained (Figure 2).

Data Models & Inheritance. The sparse use of inheritance throughout software development is a recommended best practice among developers. It minimizes assumptions about common features of classes and their roles. In Gria2-Viewer, we utilize object-oriented inheritance only to describe data models for parsing and holding *Gria2* data files (Figure 3).

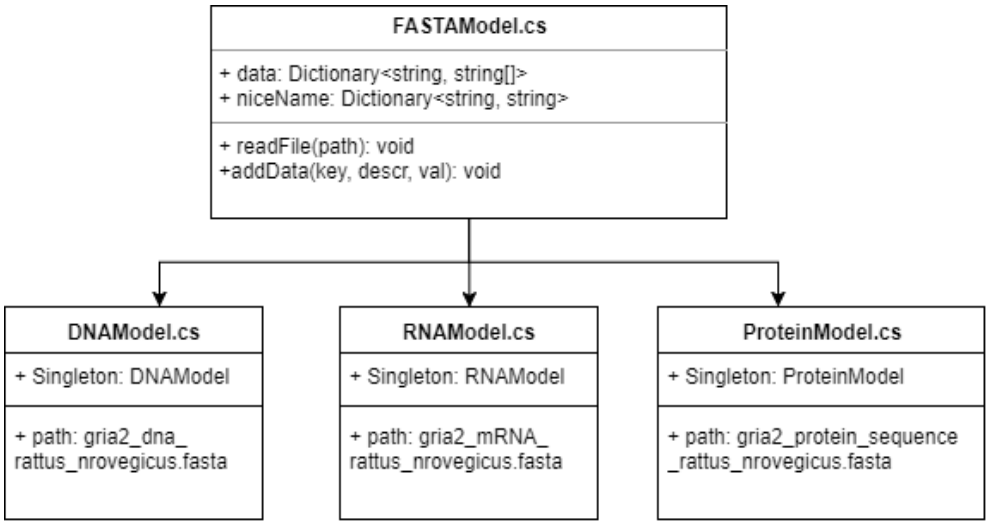


Figure 2: Inheritance is the best option to describe three different data models that rely on parsing the same file type. The niceName field maps common species names e.g. "Homo sapiens" to different key indices within the data Dictionary depending on the specific child class.

All FASTA files are processed the same way; therefore the common parsing code resides within FASTAModel.cs. Each species for which FASTA files are available map to a unique niceName

string instance, e.g. string niceName = “Homo sapiens”. The niceName string instance is common to the DNA, RNA, and Protein models for each species, but they map to different-valued keys within the data dictionary, depending upon which type of model (DNA, RNA, or Protein) the user is interested (Supp. Table 2).

UI. Hover UI Kit is a free software package available for download from github. It is governed by a GPLv3 license which authorizes free use for open source projects.

Once Hover UI libraries are imported into the project directory, an empty GameObject is created, and a Hover UI creation script component is attached. Running the script directly in Unity editor mode results in a static menu set, if given appropriate parameters. The static menu instance is directed to find and attach itself to an instance of Leap Motion hands at runtime (Figure 3). The left hand transform acts as the parent to the UI menu while the right hand acts as a pointer. The complete menu hierarchy as implemented in the current version is listed in Table 3.

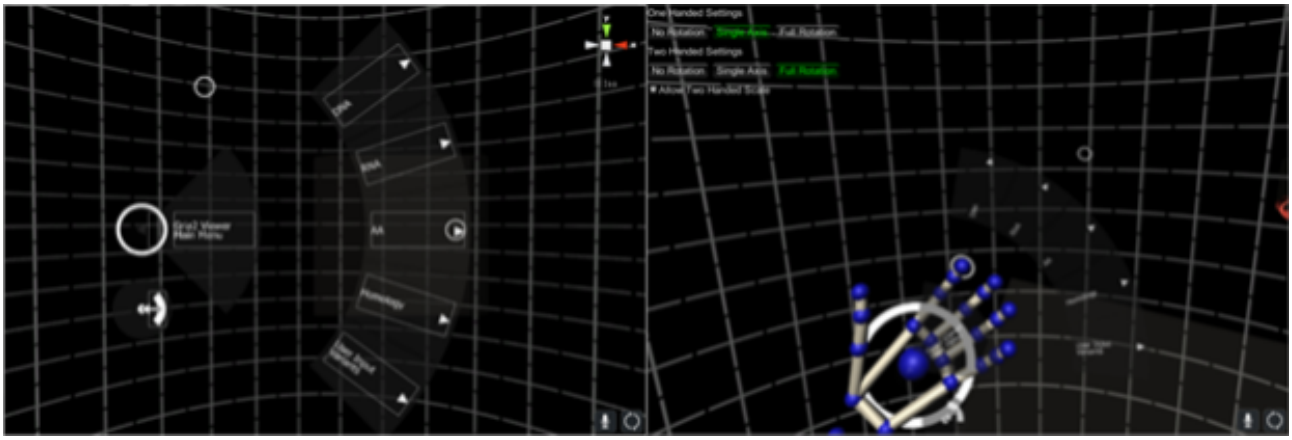


Figure 3: Hover UI v 2.0.0-alpha used in conjunction with Leap Motion Orion v3.0.0. The menu is anchored to the user’s left hand. The index finger on the right hand acts as a cursor: it generates a button pressed event for a particular button when it hovers over that button within a set amount of time. The specific timing varies and can be set per button.

- Gria2-Viewer Main Menu
  - DNA
    - Show
    - *Coding Region (CDS)*
    - Back

- RNA
  - Show
  - *Consensus (MSA)*
  - Back
- AA
  - Show on Model
  - *Consensus*
  - Back
- Homology
  - Rattus norvegicus
  - *Homo sapiens*
  - *Pan troglodytes*
  - Back
- User Input Variants
  - *Variant 1*
  - *Variant 2*
  - *Variant 3*

Table 3: The complete hierarchical menu set is an instance of Hover UI, and open source project for VR interfaces. Italics represent buttons which will be implemented in a future release.

Plane Geometry and UV Coordinates. We use plane geometry and uv coordinates to map nucleotide sequences onto UI elements. The following is an overview of Unity's plane geometry and texture mapping properties which allows for this unique sequencing representation.

A Unity plane geometry is a flat surface as defined by variables contained in its mesh instance. Every mesh instance has an array of Vector2 uv coordinates (Figure 1) which define UV the mapping of a two-dimensional texture image onto the projected surface of any valid geometry. In the case of plane geometry, uv coordinates map perfectly to the length and width of the plane such that in the default case, the u coordinate ranges from (0, 1) and spans the plane's length, while the v coordinate ranges from (0, 1) and spans the plane's width.

To render nucleotide sequences onto textures, we take advantage of two properties. First, geometries have the unique property of being allowed to partially map to uv coordinates. In other words, mesh geometries do not need span the entire uv range. Second, Unity allows the procedural editing of textures via Texture2d.SetPixel(int x, int y, Color color) where x, y refers to a texture coordinate [8]. Note that a 256 x 256 texture will map to a 1x1 uv square such that

(0,0) => (0,0) and (256, 256) => (1,1). Thus, each nucleotide within a sequence can be represented by their traditional colors (Table 4) and associated with a specific (u, v) coordinate.

Nucleotide	Color	RGB
A	Blue	68,155,255
T / U	Yellow	244,220,110
C	Red	224,81,62
G	Green	83,209,131

Table 4: Nucleotides in DNA and RNA Panels are represented by the colors listed above. Each pixel can be set to a unique color using the Texture2d.SetPixel function. Thus, a sequence of *n* nucleotides will generate *n / textureWidth* rows of *textureWidth* color squares. Each color square represents the nucleotide at a unique sequence position.

Since geometries don't need to map to the entire uv space, the MeshRenderer component of the plane geometry mesh will then only render portions of the nucleotide sequence at a time (Figure 5). The range of the nucleotide sequence to be rendered can be adjusted via scrolling. In a future version, we plan to support the rendering of a nucleotide region defined by typing in the position within the genome.

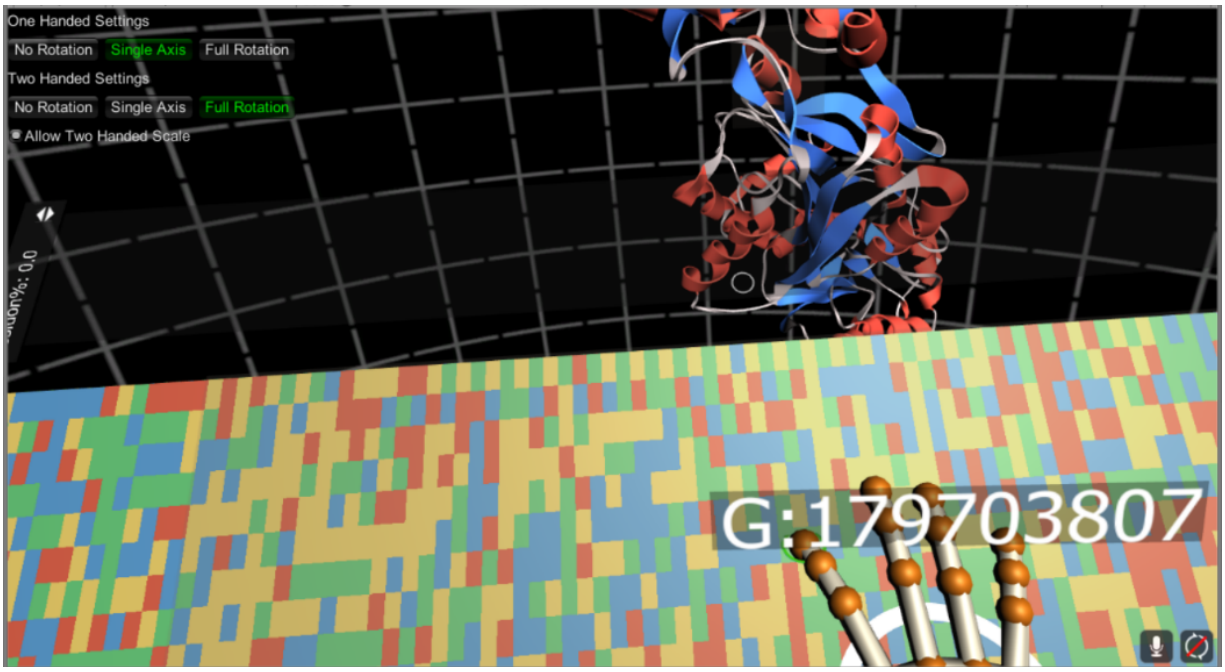


Figure 4: 120,327 base pairs of *Gria2* DNA from Rattus norvegicus are loaded into the DNA Panel shown above. As the user's right index finger hovers over a point on the panel, the base pair for that point is retrieved; it is displayed along with the position within the fasta file in which the base pair occurs.

Algorithms. MSA algorithm and global pairwise alignment are implemented within VRModel.SequenceModel, VRModel.Algorithms.SequenceAligner. Algorithms are not the focus of this paper, but it should be noted that the incorporation of algorithms allows the user to judge conservation, categorize sequences as wild-type vs variant, and locate intron / exon regions in the DNA view.

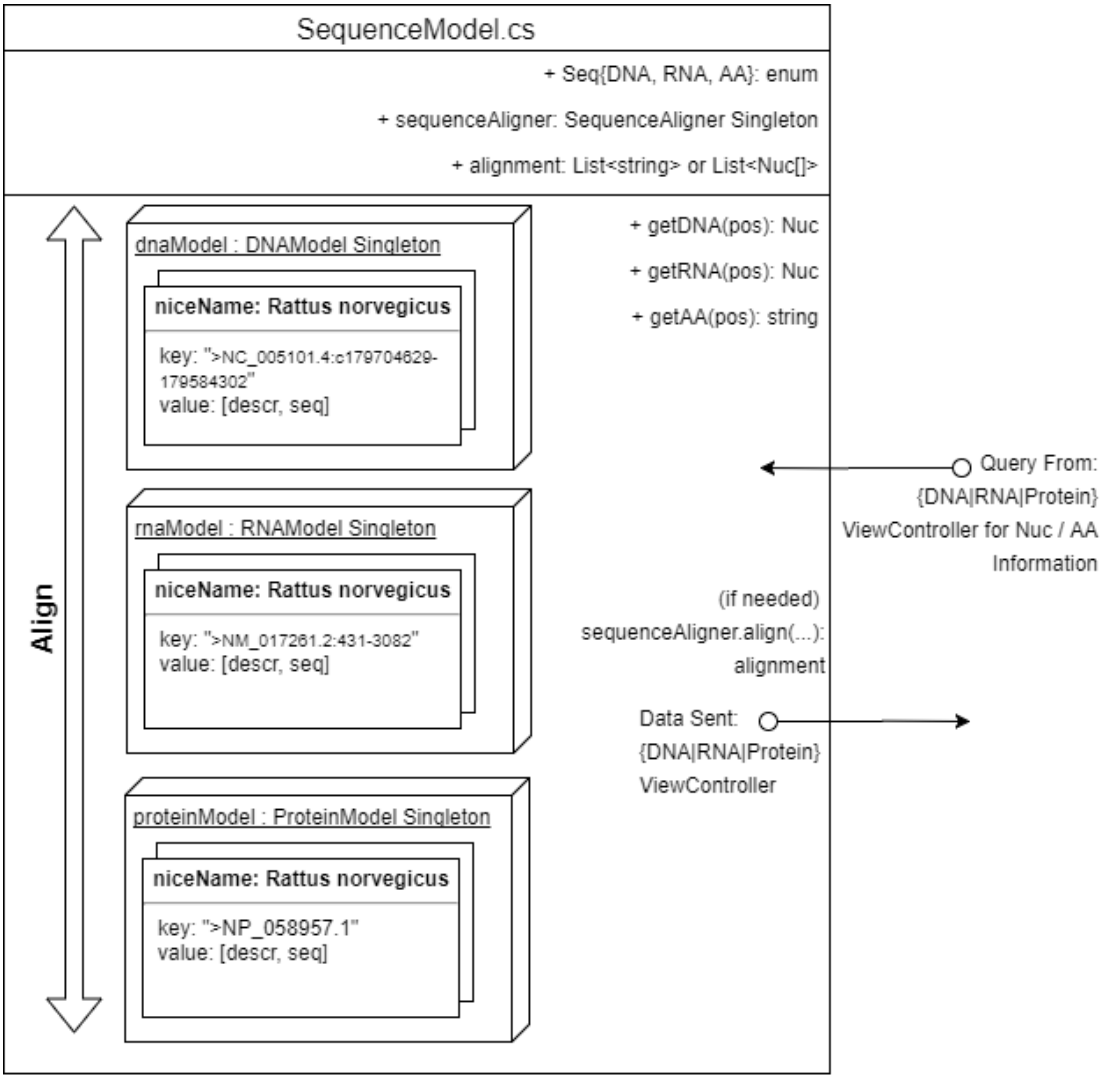


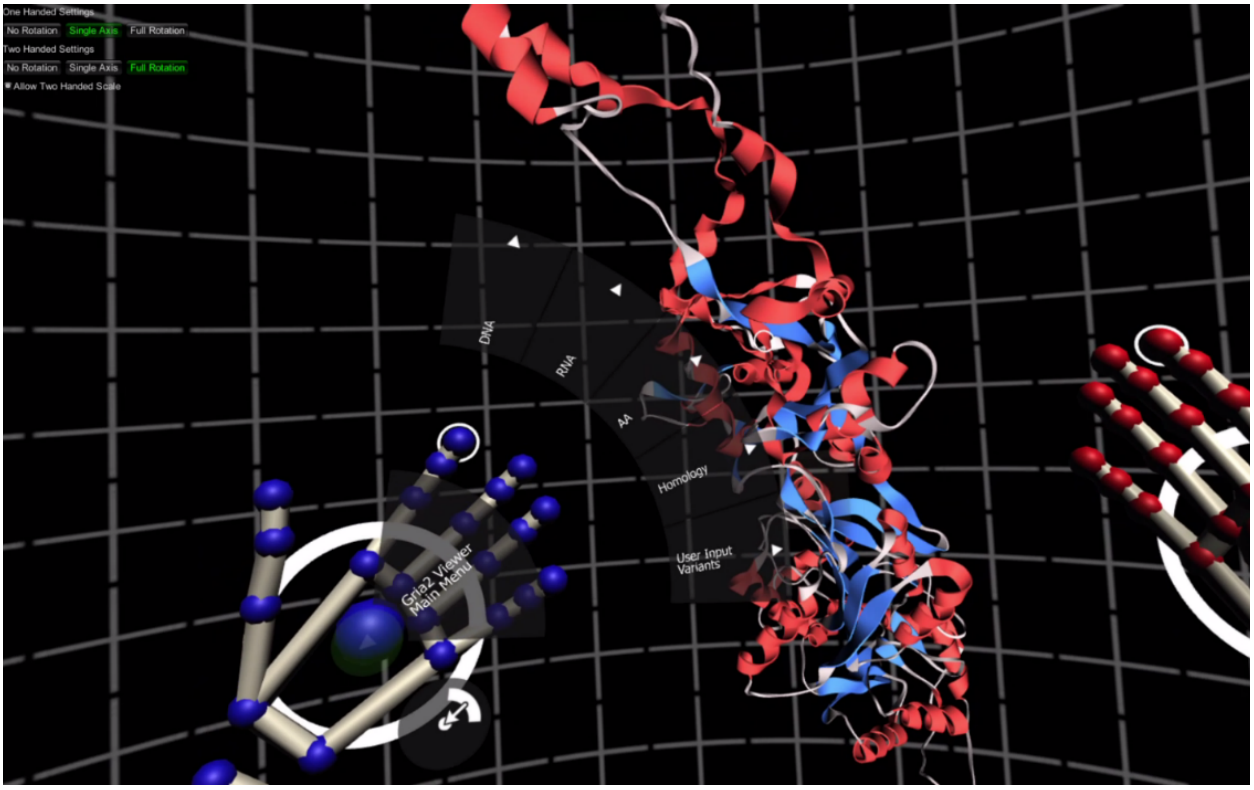
Figure 5: SequenceModel.cs keeps references to DNAModel, RNAModel, ProteinSeqModel singletons, and a reference to a SequenceAligner instance which currently implements the Needleman Wunsch pairwise alignment algorithm for global alignment.

Results

VR Software: Gria2-Viewer

The Gria2-Viewer allows scientists to view genomic, transcriptomic, and proteomic data for *Gria2* in unison. It features a simple user interface for viewing genomic information which utilizes Leap Motion tracking to turn fingers into data manipulators. To the right hand is attached an instance of Hover UI (

Figure 6: To navigate within VR, Gria2-Viewer uses a menu set anchored to the user's left hand. The menu set is an instance of Hover UI, an open source project found at https://github.com/aestheticinteractive/Hover-UI-Kit.



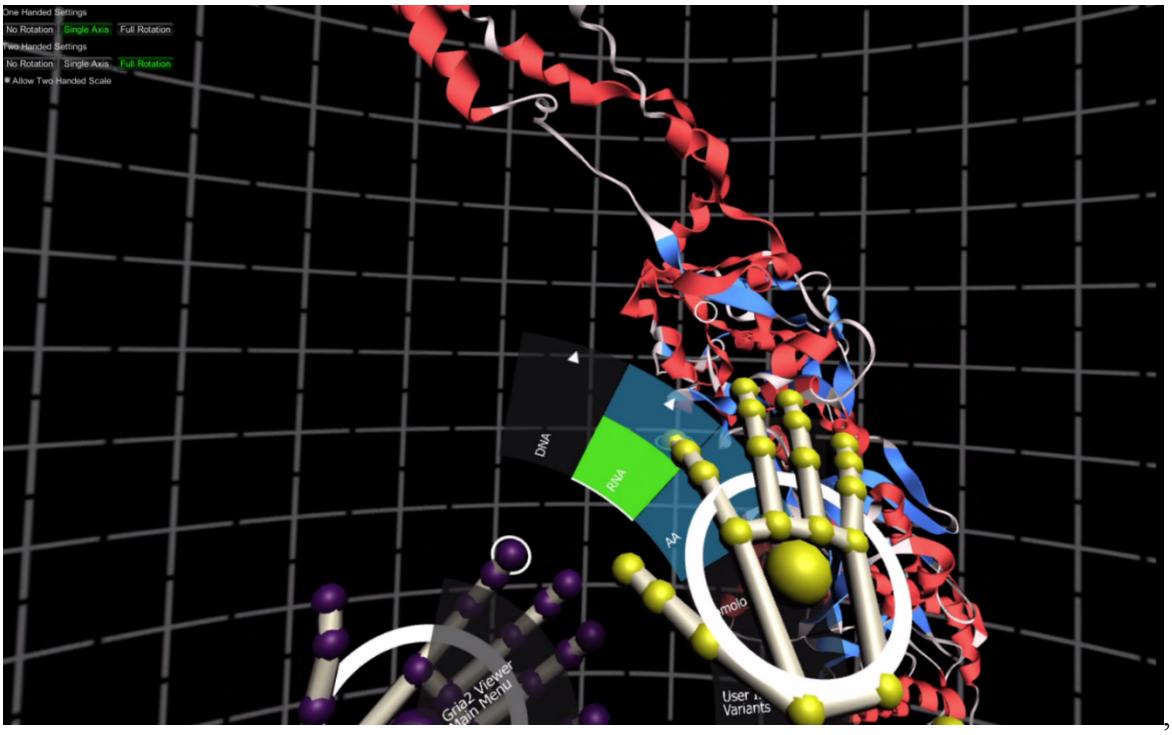
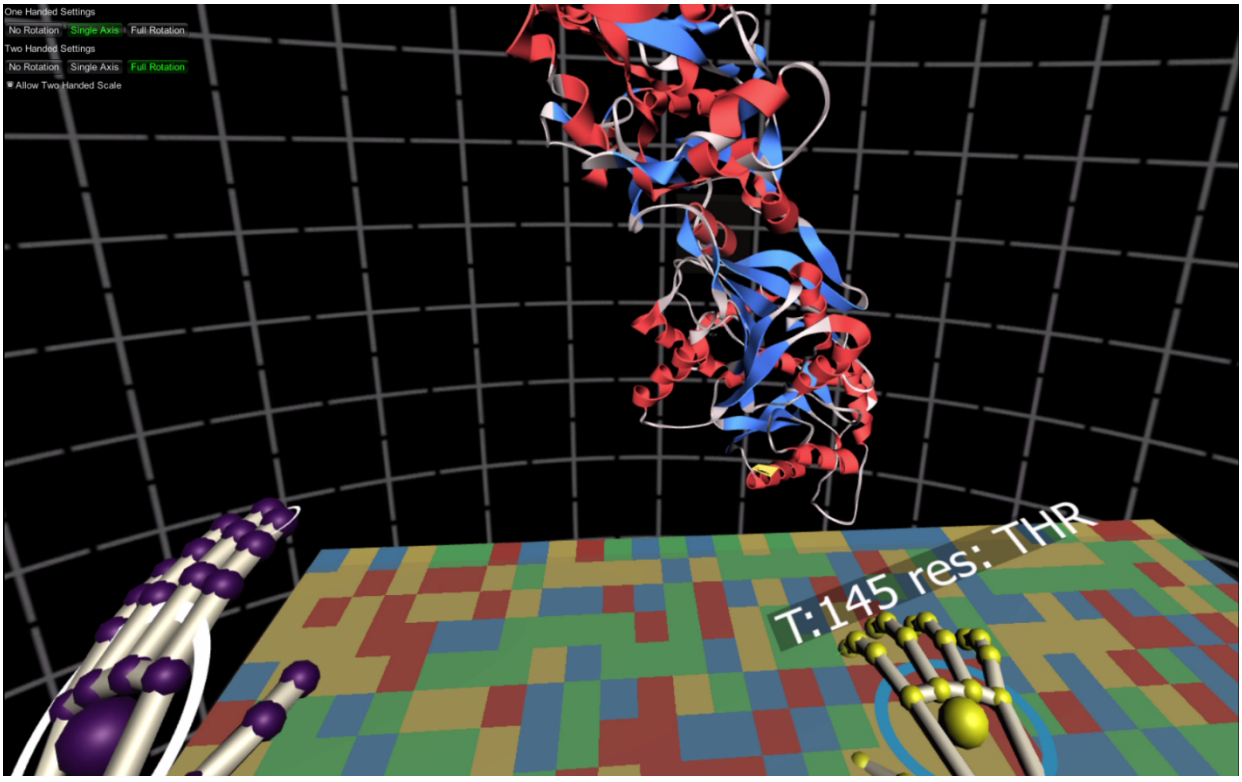


Figure 7: The menu is hierarchical and selectable with the user's right index finger or the user's Look cursor.



). The user may select one or more of the five species, and may select to view DNA or mature

mRNA nucleotides. The protein sequence in one letter code can be overlaid on top of the mRNA panel (Figure 8).

Figure 6: To navigate within VR, Gria2-Viewer uses a menu set anchored to the user's left hand. The menu set is an instance of Hover UI, an open source project found at https://github.com/aestheticinteractive/Hover-UI-Kit.

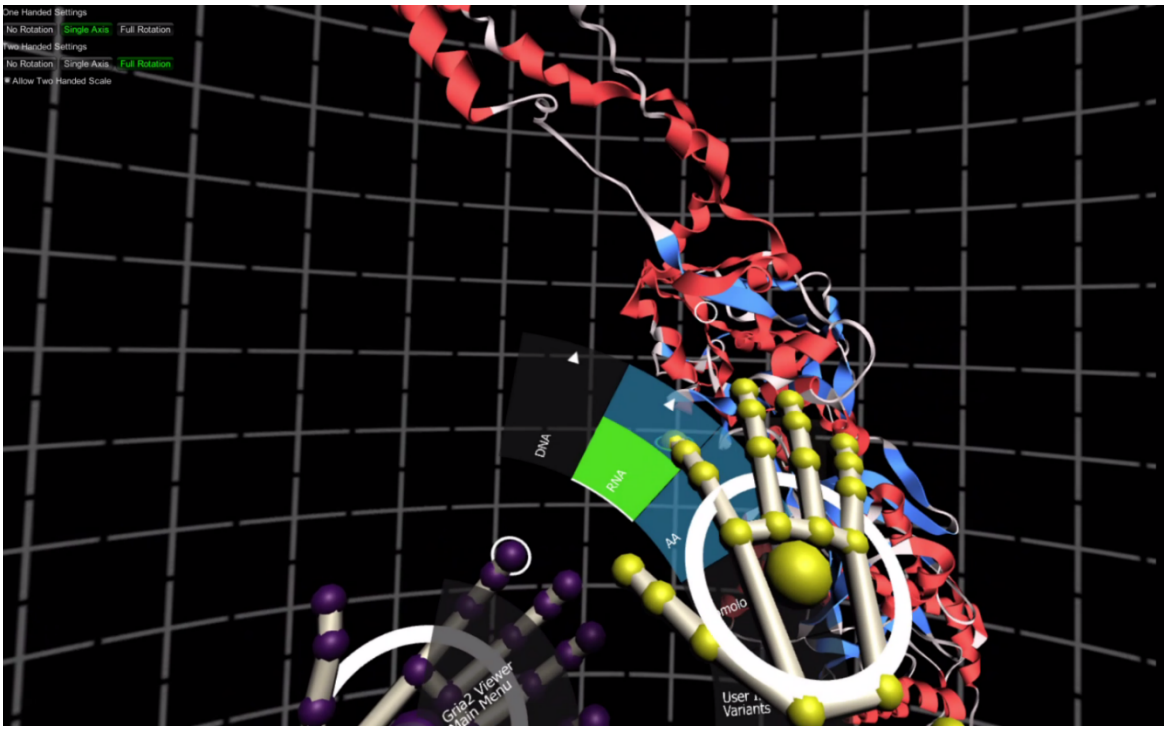


Figure 7: The menu is hierarchical and selectable with the user's right index finger or the user's Look cursor.

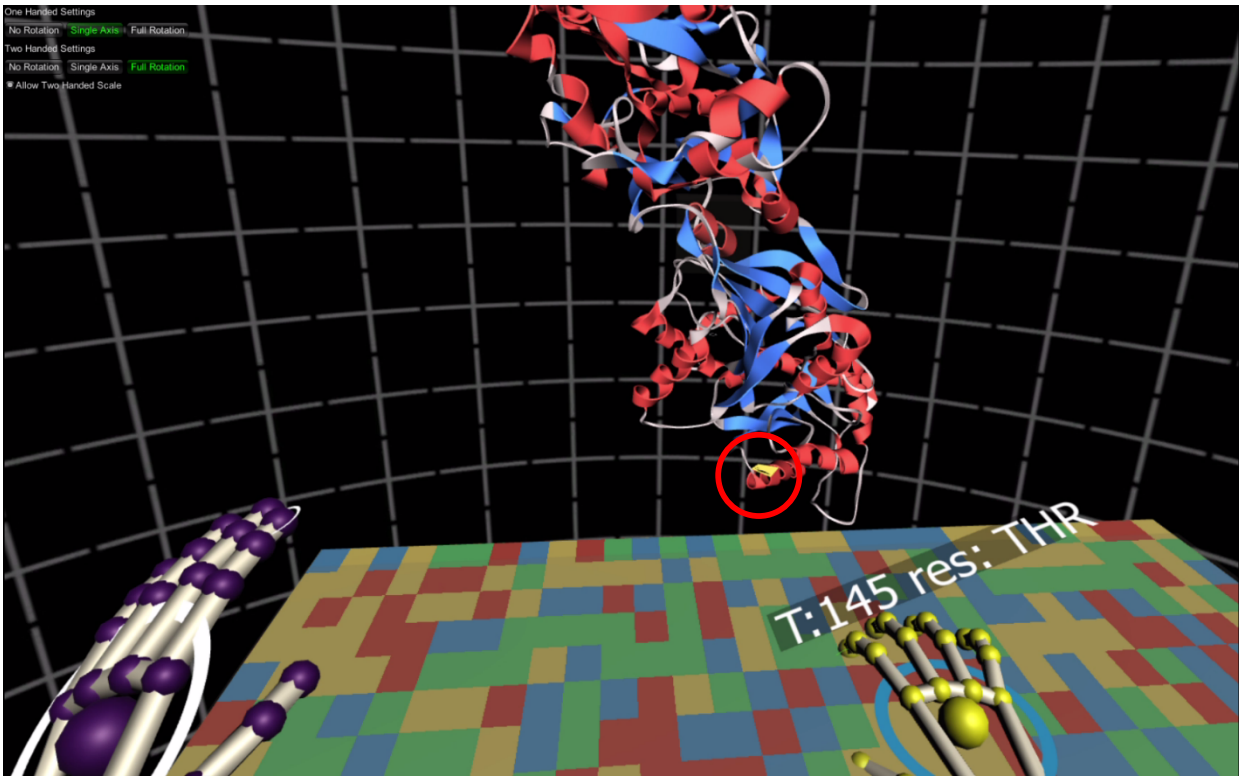


Figure 8: Viewing the *Gria2* RNA sequence for Rattus norvegicus. Above the right index finger is a GUI which displays context-sensitive data depending on where the user places his or her index finger along the sequence. The residue corresponding to the nucleotide is also highlighted in the 3d structure via a yellow outline shader (denoted by red circle). Note that the red circle was added for emphasis in this paper and does not appear in the actual program.

Nucleotide Sequences and UV Coordinates. The ideal representation of large (~100 Kb) nucleotide sequences is an open problem in both VR as well as web interfaces. We created a plane geometry with customized uv coordinates that allow for the representation of nucleotide sequences of up to 100 Kb (Figure 4). Using the BuildTexture() method in {DNA | RNA}PanelController, the appropriate fasta file is accessed and its nucleotide sequence is processed such that each texture coordinate of the DNA or RNA plane takes on a color that represents a specific nucleotide in the fasta sequence (Table 4). If the user selects “AA>Show on Model” and goes to “RNA>Show” Gria2-Viewer gives additional context. Specifically, it displays the amino acid of the 3d structure which pairwise aligns with the translated version of the displayed mRNA sequence, and its position within the secondary structure (

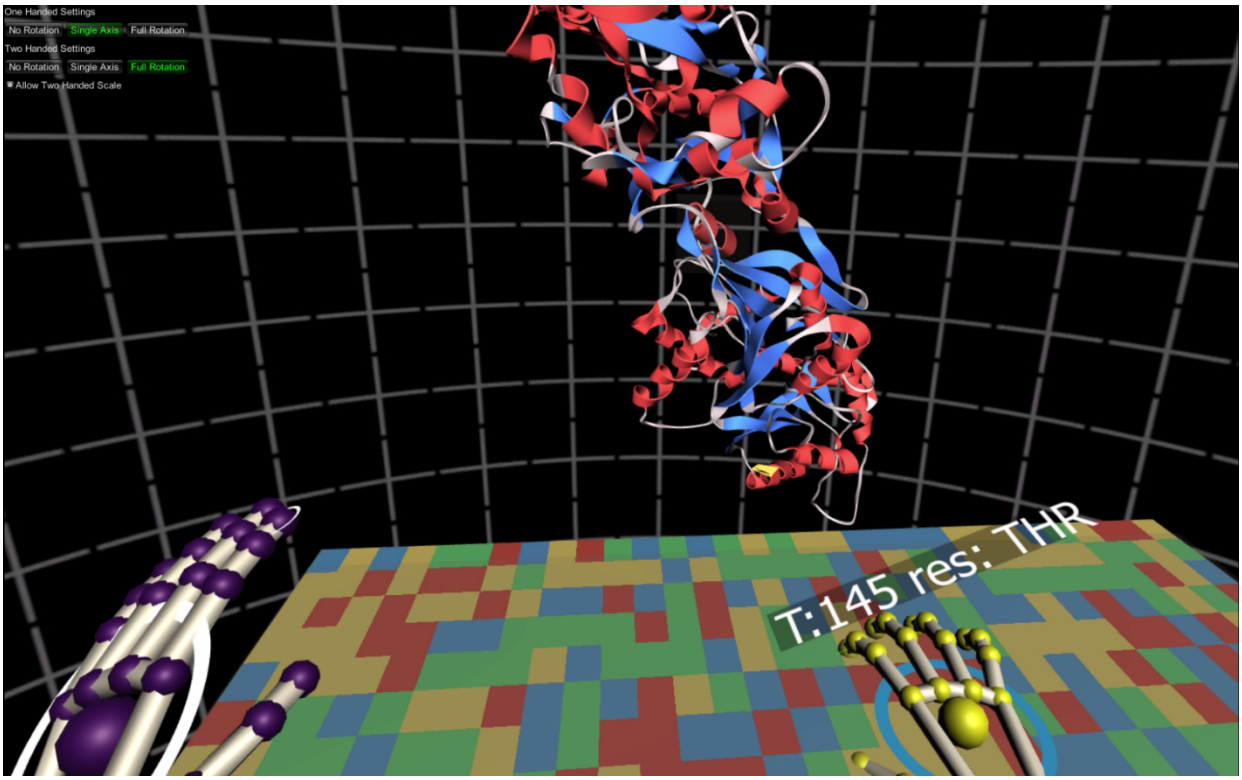


Figure 8).

User Interface Case Study

We designed a UI case study around the VR experience in hopes of comparing its effectiveness against the traditional bioinformatics workflow. Usability testing is a key process within software development. We hope to bring that process into wider adoption within the bioinformatics community by introducing this case study. Key issues involving the details of the case study remain, such as which metrics should be monitored, and what statistical results constitute a measure of "effectiveness." Nevertheless we hope these issues will be resolved in future versions of the Case Study manual.

User Interface Case Study : VR vs Traditional Methods for Analysis of Gria2 Variants

Study Authors: Jimmy Fan Zhang, Feng Cui Ph.D., Gary Skuse Ph.D., Paul A. Craig Ph.D, Joe Geigel, Alex Paciorkowski MD

Introduction

The increasing popularity and usefulness of Virtual Reality (VR) applications requires an empirical study of their scientific utility within the bioinformatics context. We propose a research study comparing a VR application (VR arm) against existing methods (Traditional arm) in determining various attributes of *Gria2* genomic variants. The VR application is suited to viewing *Gria2* genomic, transcriptomic, and proteomic data. The traditional method consists of an internet-connected computer with access to the same *Gria2* datasets, sans VR capability. We plan to recruit bioinformatics students within RIT campus; qualified subjects will answer questions about real-life *Gria2* variants identified through exome sequencing from Paciorkowski Lab. Subjects will be provided technologies from one of the two aforementioned study arms: VR vs Traditional. Subjects will be given as much time as required, but will be timed without their knowledge in their efforts to answer the questions. The study has two objectives: 1.) to determine whether some differential time interval exists between the two study arms for completing the *Gria2* questions, and 2.) whether the correctness of responses to the questions is significantly different among the two arms.

Gria2 Background

Gria2 is a gene associated with intellectual disability (ID) [10]. It encodes a glutamate receptor subunit called glutamate receptor 2. Glutamate receptors are highly expressed in postsynaptic structures and has an important role in signal transduction across synapses in the central nervous

system. This specific class of glutamate receptors are considered ligand-gated ionotropic transmembrane receptors for glutamate. Any nonsynonymous sequence variations which affect the *Gria2* protein structure either along its ligand interface, or disrupts its ability to tetramerize with Gria1-4 to form the complete glutamate receptor, will negatively impact the process of membrane depolarization required for neurotransmitter release. The Gria2 gene sequence is available on NCBI. Its protein sequence data are available on PDB.

Safety & Ethics

The study will be conducted in compliance with standards established by the Human Subjects Research Office (HSRO) and Institutional Review Board (IRB) at Rochester Institute of Technology (RIT).

Subject written consent will be obtained after investigators explain the study and before any study-related procedures occur. Health risks associated with prolonged VR use will be explained. Additional care will be given to subjects who might require any kind of special medical accommodations, interpreting services, etc. with the exception that visually impaired subjects are discouraged from study participation for safety / health reasons. Subjects may choose to not participate or withdraw from the study at any time for any reason.

Study Objectives

The study aims to determine whether subjects randomized to the VR arm perform better than subjects randomized to the Traditional arm at completing the given problem set. To statistically quantify “perform better” we consider the following metrics: the average time interval among the two arms for completing the *Gria2* questions, and the correctness of responses to the questions.

Investigational Plan

Recruited subjects who qualify are randomized into one of two arms (VR or Traditional) and provided with four pieces of sequencing data: two belonging to wild-type *Gria2*, the other two are disease-causing *Gria2* variants found by Paciorkowski et al. Each of the wild-type sequences will span a region covering one of the two variants. Subjects are then given questions about the data and are asked to solve them according to the tools provided by their study arm. The problem is as follows:

> Given an intron-free DNA sequence *x* that resides within the *Gria2* locus, determine if *x* represents:
>
> - a wild-type sequence. If so, move on to the next sequence.
> - a mutation (note that there are no frameshift or nonsense mutations). If so, answer the following:
>   - What kind of mutation is it, synonymous or non-synonymous?
>   - How damaging is the substitution? (Synonymous = no damage, some damage, or very damaging)
>   - Which specific amino acids are affected, if any? (Give loci and the amino acid change)
>   - highlight the position of the amino acid in the *Gria2* 3D structure.

The following metrics are then collected:

- Time elapsed, as determined from the moment the subject accesses the above questions, to the moment they turn in their work.
- The correctness of the submitted work.

- Whether the subject ended work without completing, and if so, at what point during the workflow did they give up.

Subject Recruitment Population

Undergraduate Bioinformatics students from RIT are encouraged to participate in this study. Upon recruitment, a visual questionnaire and a mental health questionnaire are given to subjects. We hope to conduct the Case Study on subjects who score well on the vision and mental health questionnaires. The purpose of these restrictions is two-fold: 1.) to prevent health complications arising from poor ocular health coupled with the VR experience, and 2.) to reduce undue harm and stress arising from attempting to complete the tasks associated with the study.

Statistical Methods

The proportion of subjects who did not complete the questions for each study arm will be noted. The scoring of partial answers from these subjects is to be determined at a future date. The test statistic for this type of data is pending.

The time elapsed metric will be analyzes using the student's t-test since it is expected to follow the t-distribution under our hypothesis.

Methods for scoring the correctness of the answers is an area of future work. The problem begins with placing the sequence x into one of two categories, suggesting a chi-square test. If x falls into the latter category, the next four questions require a mixture of categorical (synonymous vs nonsynonymous), ordinal (severity category) and continuous (any point along the *Gria2* ribbon) answers. Test statistics on these questions might require a mixture of chi-squared, Wilcoxon, and student's t-tests.

These tests will be compared in aggregate to determine the overall effectiveness of one study arm compared with the other, pending further development.

Discussion

UCSC's Genome Browser, Ensembl Project, and Integrative Genomics Viewer (IGV) represent the main genome browsers available to research scientists. In proteomics, UCSF Chimera, PyMol, and JMol are available choices. As of this writing, there is a lack of information on the relative popularity of these tools. This makes comparisons among the various software offerings and their respective UI paradigms difficult to achieve. This is problematic for the following reason.

In informatics and biology, new algorithms or experimental procedures gain widespread acclaim if they lead to orders of magnitude increase in research productivity. Through everyday use of computers for both casual and professional purposes, we intuit that user interfaces have a tangible influence on efficiency. Milestones within the technology industry are often marked by major user interface innovations. From the mouse to Windows 95 to touchscreen interfaces, UI has always had profound an impact on user growth and consumer adoption.

Efficiency gains in genomic analysis may occur during the data visualization stage. Without a case study involving potential software users, there is no metric for determining these kinds of efficiency gains. Neglecting the improvement, testing, and maintenance of user interfaces in bioinformatics tools is analogous to presupposing that such improvements are negligible. To the contrary, we believe that user interfaces are critical towards understanding bioinformatics datasets. The following are current problems in bioinformatics for which VR-enabled software may have significant potential.

Chromatin Modeling. In interphase, uncondensed chromatin takes on three-dimensional structures within the nucleus that may facilitate long range interactions among distant (>2kb) loci. Lieberman-Aiden et al. developed a technique to map conformations of whole genomes. They show the approximate spatial proximity of loci at 1 Mb resolution and propose a fractal globule structure to explain their chromatin conformation [11]. The fractal globule structure can be modeled by a Hamiltonian walk: every point is visited once and no paths intersect. The structure plays an important role in determining loci adjacencies.

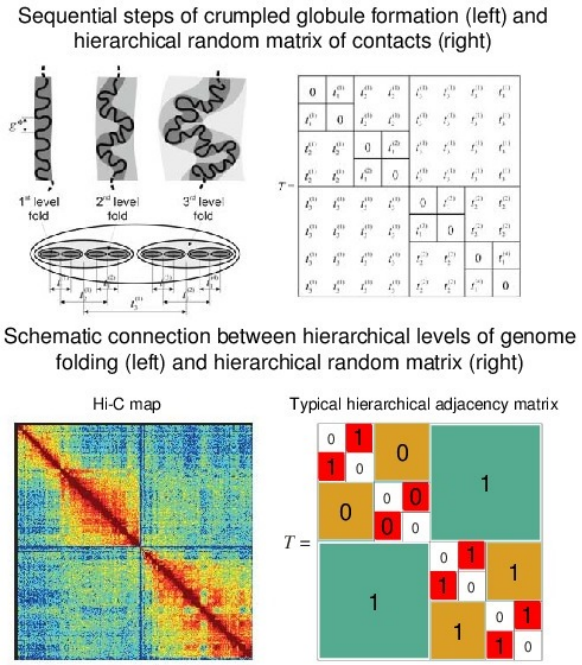


Figure 9: Researchers use adjacency matrices to model spacial relations between distant loci. Color values encapsulate rgb-values but are commonly utilized in within a one-dimensional spectrum. Thus, each cell in an adjacency matrix essentially contains 1-dimensional data. Figure adapted from Le Laboratoire de Physique Théorique et Modèles Statistiques (LPTMS).

An adjacency matrix is a popular method of representing fractal globules, with the caveat that any data type represented at the coordinate *(x,y)* in the matrix is necessarily one-dimensional by nature. Geometric attributes from adjacency data must be inferred. This is an ongoing topic of research [12]; algorithms and statistical models have recently been developed to help estimate topology from co-localization data [13, 14]. Future algorithmic or statistical developments could

couple cells within adjacency matrices with multiple types of datasets. The potential to render multiple dimensions of data at a given spatial location is something for which adjacency matrices are ill-suited.

Epigenetics. Despite recent advances in epigenome research, our understanding of epigenetic markers and their roles in regulation is similarly hindered by the inability to visualize their impact across the genome. The relationship between epigenetic patterns to genomic loci, gene expression, and phenotype remains challenging to visualize. A 2007 publication from Professor Tamassia of Brown University outlines the visualization requirements of depicting various biological systems. He notes that recent advances in high throughput sequencing technology has exacerbated the need to develop visual exploration techniques to enhance researchers' ability to deduct meaningful biological information [15].

Temporal Biological Networks. Depictions of metabolomics networks, and biological pathways present unique visualization challenges. A current limitation of pathway depictions in journal articles is the static display of inherently temporal datasets.

In a trivial example, the eukaryotic cell cycle is often depicted as a large circle with numerous checkpoints. For researchers interested in the cell cycle as influenced by the fluctuation of cellular signals over time, the standard depiction loses its importance. One could argue that time-series data such as p53 expression may be plotted around the outer circumference of the circle, but that is not necessarily a good solution. If the researcher then wants to consider p53 regulation by its promoters or inhibitors, he or she must plot the time-series data for those markers as well. Conversely, if the researcher is done contemplating the role of p53 and wants to consider the fluctuation of another biomarker, he or she must generate another plot.

An Omic VR application can be well-suited to the problems described above. Whole genome representation can be rendered at low resolution until the user decides to investigate a gene locus, upon which the VR application may zoom in on the region of interest at higher resolutions. Interactions among promoters, enhancers, and silencers at the loci are shown depending on context. Geometric topologies within chromatin regions are made obvious to the user. Finally, animated simulations can help users visualize temporal datasets.

Future Work

Loading Other Proteins

Except for residue highlighting on secondary structures, a modified version of Gria2-Viewer performs the same functions on the voltage-dependent L-type calcium channel subunit beta-2 (Uniprot KB Q8VGC3, CACNB2 gene). The pdb file associated with the protein, 5v2p.pdb, should be trimmed of HETATM and ANISOU entries. We suspect that the issue is the result of a bug in Splitting.cs. We anticipate further development of the project throughout the year, with a more polished version that is capable of fully supporting proteins other than Gria2 and a BMS publication submission by the end of 2017. The remaining features we plan to support are as follows:

- Upon user activation of the “Consensus” button within the DNA, RNA, or Protein menus, consensus regions populate the panel and are denoted in two ways. First, a transparency attribute associated with each color is applied to the panel.
- The UI text attached to the right hand that shows which homologous sequences from the species enabled under “Homology” contributed to the consensus at the position indicated by the right index finger.

- The “User Input Variants” menu item loads user-defined experimental data. In the Case Study we preload the submenu with wildtype and variants found from Paciorkowski Lab.

Conclusions

Virtual reality is a ground-breaking medium with major advantages over traditional visualization for bioinformatics datasets. Its potential remains largely unexplored. We show that genomic, transcriptomic and proteomic data for Gria2 can be viewed in aggregate within VR, leading to a novel workflow for researchers. We also show that results of pairwise and MSA alignment algorithms can be rendered onto VR geometry as textures in UV space. Finally, we designed a user case study to compare the VR workflow against the traditional workflow consisting of a text editor and access to bioinformatics websites.

References

1. **Unity 3d: One Game Engine to Rule Them All** [https://digit.hbs.org/submission/unity-3d-one-game-engine-to-rule-them-all/]
2. Perez S, Tubiana T, Imberty A, Baaden M: **Three-dimensional representations of complex carbohydrates and polysaccharides--SweetUnityMol: a video game-based computer graphic software**. *Glycobiology* 2015, **25**(5):483-491.
3. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourne PE: **The Protein Data Bank**. *Nucleic Acids Research* 2000, **28**(1):235-242.
4. Okada O, Odai K, Sugimoto T, Ito E: **Molecular dynamics simulations for glutamate-binding and cleft-closing processes of the ligand-binding domain of GluR2**. *Biophys Chem* 2012, **162**:35-44.
5. Yelshanskaya MV, Singh AK, Sampson JM, Narangoda C, Kurnikova M, Sobolevsky AI: **Structural Bases of Noncompetitive Inhibition of AMPA-Subtype Ionotropic Glutamate Receptors by Antiepileptic Drugs**. *Neuron* 2016, **91**(6):1305-1315.
6. Jimmy Zhang TK, Emily Tuttle, Dalia Ghoneim, Laurie E Seltzer, Inna Hughes, David Wang, Liu Lin Thio, Kristen Lindstrom, Christina Gurnett, Argirios Neinopoulos, William B. Dobyns, ?Matheus Malta de Sá?, Laura Silveira Moriyama, Alex Paciorkowski: **Novel genetic variants identified by massively parallel sequencing in infantile spasms**. Pending.
7. **National Center for Biotechnology Information (NCBI)**
8. Unity Technologies: **Unity - Scripting API**. 2017(2017.1).
9. Raim J: **Finite State Machines in Games**. In.: Lehigh University; 2005.
10. Hackmann K, Matko S, Gerlach E-M, von der Hagen M, Klink B, Schrock E, Rump A, Di Donato N: **Partial deletion of GLRB and GRIA2 in a patient with intellectual disability**. *European Journal of Human Genetics* 2013, **21**(1):112-114.
11. Lieberman-Aiden E, van Berkum NL, Williams L, Imakaev M, Ragoczy T, Telling A, Amit I, Lajoie BR, Sabo PJ, Dorschner MO *et al*: **Comprehensive Mapping of Long-Range Interactions Reveals Folding Principles of the Human Genome**. *Science* 2009, **326**(5950):289.
12. Mirny LA: **The fractal globule as a model of chromatin architecture in the cell**. *Chromosome Res* 2011, **19**(1):37-51.
13. Moscalets AP, Nazarov LI, Tamm MV: **Towards a robust algorithm to determine topological domains from colocalization data**. *ArXiv e-prints* 2016, **1601**:arXiv:1601.01253.
14. Varoquaux N, Ay F, Noble WS, Vert JP: **A statistical approach for inferring the 3D structure of the genome**. *Bioinformatics* 2014, **30**(12):i26-33.
15. Tamassia R: **Handbook of Graph Drawing and Visualization (Discrete Mathematics and Its Applications)**: Chapman \\& Hall/CRC; 2007.