Technical University of Košice

Faculty of Mining, Ecology, Process Control and Geotechnologies

Application of Virtual and Augmented Reality in Education

Master's Thesis

2017 Bc. Michal Takáč

Technical University of Košice

Faculty of Mining, Ecology, Process Control and Geotechnologies

Application of Virtual and Augmented Reality in Education

Master's Thesis

Study Programme:	Informatization Process of Obtaining and Processing Raw Materials
Field of study:	Gathering and Processing of Earth Resources
Department:	Institute of Control and Informatization of Production Processes (ÚRaIVP)
Supervisor:	RNDr. Andrea Mojžišová, PhD.
Consultant(s):	RNDr. Jana Pócsová, PhD.

Košice 2017 **Bc. Michal Takáč**

Abstract

In this thesis we set to explore the possibilities of virtual and augmented reality (VR/AR) through the use of modern web technologies and development practices with the goal of finding novel approaches and applications in higher education with focus on mathematics. First a brief history of VR and AR is described followed by overview regarding the current problems and use cases of VR and AR systems and overview of current commercially available headsets. Then the novel, experimental Graphical User Interface (GUI) is presented, utilizing six degrees of freedom in room-scale virtual world within web page with use of interactions with VR hand controlelrs to provide strong visual representations and visualizations of parametrized functions, helping students to approach learning differently and understand difficult math concepts faster. In the end of thesis we propose future possibilities and how the technology could shape student's knowledge.

Keywords

Virtual reality, Augmented reality, Education, Mathematics

Assign Thesis

Namiesto tejto strany vložte naskenované zadanie úlohy. Odporúčame skenovať s rozlíšením 200 až 300 dpi, čierno-bielo! V jednej vytlačenej ZP musí byť vložený originál zadávacieho listu!

Declaration

I hereby declare that this thesis is my own work and effort. Where other sources of information have been used, they have been acknowledged.

Košice, April 24, 2017 ....

Signature

Acknowledgement

I would like to express my sincere thanks to RNDr Andrea Mojžišová, PhD, the main Supervisor, for her constant and constructive guidance throughout the study, valuable feedback and brainstorming sessions. Special mention should go to RNDr Jana Pócsová, PhD. for her interest in new approaches to teaching and her ability to listen and give feedback to novel ideas when others are not interested and also to prof. Steven Abbott for helping me out with Oculus Touch support. To all other who gave a hand, I say thank you very much.

Preface

Mathematical knowledge is often fundamental when solving real life problems. Especially, problems situated in the three-dimensional domain that require spatial skills are sometimes hard to understand for students. Many students have difficulties with spatial imagination and lack spatial abilities. Recently, a number of training studies have shown the usefulness of virtual reality in training spatial ability. Therefore I set myself a goal for finding inresections between virtual reality and higher education by building experimental virtual user interface(s) and testing them in educational environments, e.g. colleges and universities. In this thesis I'll focus on mathematics.

Contents

List of Figures

List of Terms

Introduction

Our experimental project will be called MathworldVR, which sets to explore the possibilities and introduce novel methods of using web technologies for creating room-scale, immersive learning environment in virtual reality for helping students to explore, learn about and experiment with various parametrized functions. It's also a practical tool for teachers to showcase abstract concepts in concrete 3D space during lectures.

1 Problem expression

ajcdbabshcb

2 Theoretical analysis

Undoubtedly VR has attracted a lot of interest of people in last few years. Being a new paradigm of user interface it offers great benefits in many application areas. It provides an easy, powerful, intuitive way of human-computer interaction. The user can watch and manipulate the simulated environment in the same way we act in the real world, without any need to learn how the complicated (and often clumsy) user interface works. Therefore many applications like flight simulators, architectural walkthrough or data visualization systems were developed relatively fast. Later on, VR has was applied as a teleoperating and collaborative medium, and of course in the entertainment area.

One can say that virtual reality established itself in many disciplines of human activities, as a medium that allows easier perception of data or natural phenomena appearance. Therefore the education purposes seem to be the most natural ones. The intuitive presentation of construction rules (virtual Lego-set), visiting a virtual museum, virtual painting studio or virtual music playing (Loeffler, 1995) are just a few examples of possible applications.

Virtual environments are inherently three-dimensional. They can provide interactive playgrounds with a degree of interactivity that goes far beyond what is possible in reality. If using VR as a tool for mathematics education, it ideally offers an added benefit to learning in a wide range of mathematical domains (Kaufmann, 2011).

3 Analysis of current state

3.1 Building virtual reality applications and experiences

The leading platform for building VR experiences today is the game engine Unity, both because the company had the foresight to add support for the Oculus Rift development kit early on, but also simply because the early use cases from when Oculus Rift was still just a very successful Kickstarter project centered around video games.

3.2 Virtual reality on the web

WebVR provides support for exposing virtual reality devices — for example head-mounted displays like the HTC Vive or Oculus Rift — to web apps, enabling developers to translate position and movement information from the display into movement around a 3D scene in browser. As of today, support for both head-mounted displays is available in experimental or development builds of Chrome and Firefox, with official release planned for 2017. This has numerous very interesting applications, from virtual product tours and interactive training apps to immersive first person games. Unity, for instance, is able to make native builds for all major platforms from the same code base, including PC, Mac, Linux, iOS, Android and more. When made by professionals, such native builds will undoubtedly look better and run faster than a comparable VR experience built with WebGL and WebVR (at least AAA games or other experiences where high fidelity and performance are paramount).

The major advantage of WebVR over natively built experiences is the same as the web has always had over desktop apps and mobile apps today - no need to download and install anything. User just needs to click a link, type in a url, and the application runs directly in her browser. There's no app store needed. Web developers can also take advantage of many open source libraries available on the internet.

4 Technologies

4.1 Three.js

Three.js is cross-browser JavaScript library/API used to create and display animated 3D computer graphics in a web browser. It uses WebGL.

4.2 A-Frame

A-Frame is a web framework for building virtual reality experiences. It was started by Mozilla to make WebVR content creation easier, faster, and more accessible. A-Frame lets you build scenes with just HTML while having unlimited access to JavaScript, Three.js, and all existing Web APIs. It uses an entity-component-system pattern that promotes composition and extensibility. It is free and open source with a welcoming community and a thriving ecosystem of tools and components.

There are no build steps or boilerplate required nor anything to install; all we need is an HTML file:

```
<html> <head> <script src="https://aframe.io/releases/0.5.0/aframe.min.js"></script>
</head> <body> <a-scene> <a-box color="6173F4" opacity="0.8" depth="2"></a-
box> <a-sphere radius="2" src="texture.png" position="1 1 0"></a-sphere> <a-
sky color="ECECEC"></a-sky> </a-scene> </body> </html>
```

<a-scene> contains all of the objects in our 3D scene. It also handles all of the setup that is traditionally required for 3D: setting up WebGL, the canvas, camera, lights, renderer, render loop as well as out of the box VR support on platforms such as

HTC Vive, Oculus Rift, Samsung GearVR, and smartphones (Google Cardboard). Tons of repeated code eliminated with one clean line of HTML.

Then we can place objects within our scene using assorted primitive elements that come with A-Frame such as <a-box> or <a-sphere>. This is extremely readable, and we could copy and paste this HTML to any other scene and it would behave the same. And we can use the browser's DOM Inspector just as we would with for any other web site.

4.2.1 JavaScript and Aframe

We can use traditional JavaScript DOM APIs to manipulate A-Frame scenes to add logic, behavior, and functionality:

```
var box = document.querySelector('a-box'); box.getAttribute('position'); box.addEventListener('
function () box.setAttribute('color', 'red'); );
```

And being based on the DOM, most existing libraries and frameworks work beautifully on top of A-Frame such as React, Vue.js, d3.js, jQuery, or Angular. The existing web ecosystem of tools were built on top of the notion of manipulating plain HTML and are thus compatible with A-Frame.

4.2.2 Entity-Component System

A-Frame at its core is an entity-component-system framework. Entity-component-system (ECS) is a pattern popular in game development and is prominent in game engines like Unity. ECS favors composition over inheritance. Every single object in the scene is an entity. An entity is an empty placeholder object that by itself does nothing. We plug in reusable components to attach appearance, behavior, functionality. And we can mix-and-match different components and configure them in order to define different types of objects.

Object-oriented and hierarchical patterns have well-suited the 2D web, where we lay out elements and components that have fixed behavior on a web page. 3D and VR is different; there are infinite types of objects with endless complexity. We need an easy way to build up different kinds of objects without having to create a special class for each one.

In A-Frame, an entity is simply:

<a-entity></a-entity>

A-Frame components (not to be confused with Web Components) are reusable modules that can be plugged into any entity. They are allowed to do anything and have full access to JavaScript, three.js, and Web APIs. The structure of a basic component may look like:

AFRAME.registerComponent('foo', schema: bar: type: 'number', baz: type: 'string' , init: function () // Do something when component is plugged in. , update: function () // Do something when component's data is updated.);

Then once defined, we can plug this bundle of appearance, behavior, or functionality into an entity straight from an HTML attribute.

<a-entity foo="bar: 5; baz: qux"></a-entity>

4.2.3 Component Ecosystem

A-Frame ships with several components, but since A-Frame is fully extensible at its core, the community has filled the ecosystem with tons of components such as physics, particle systems, audio visualizations, and Leap Motion controls. This ecosystem is the lifeblood of A-Frame. A developer can build a component and publish it, and then someone else can take that component and use it straight from HTML without even having to know any JavaScript.

These components are curated and collected into the A-Frame Registry. This is similar to the collection of components and modules on the Unity Asset Store, but free and open source. We make sure they work well and from there they are easily searchable and installable through multiple channels. One of which is through the A-Frame Inspector.

4.2.4 A-Frame Inspector

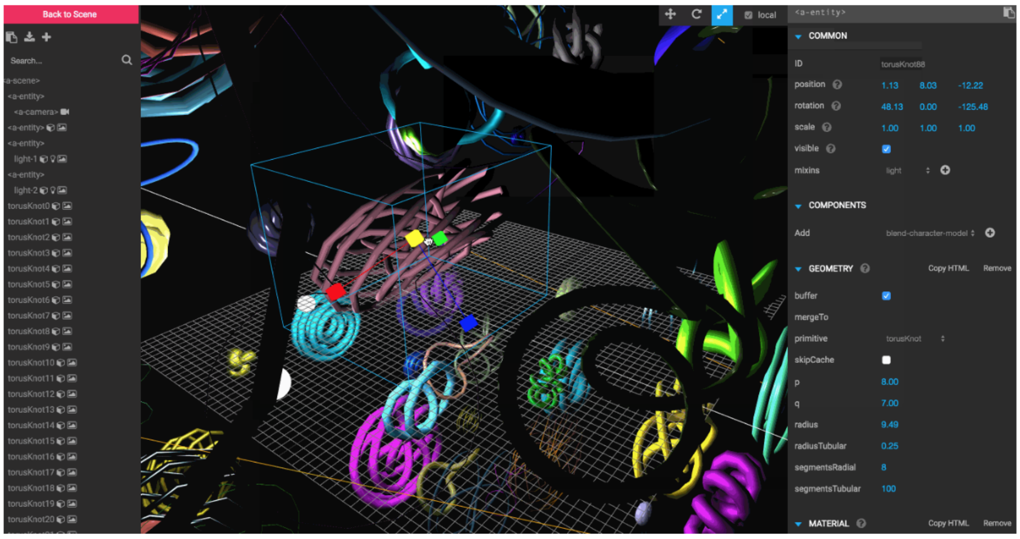
The A-Frame Inspector is a visual tool for inspecting and editing A-Frame scenes. Similar to the browser's DOM Inspector, you can go to any A-Frame scene, local or on the Web, and hit <ctrl> + <alt> + i on your keyboard.

This will open the visual Inspector where you can make changes and return to the scene with the reflected changes. You can visually move and place objects, poke around with properties of the components, or pan the camera around to see a different view of the scene. It's like viewing the source in an interactive way.

The A-Frame Inspector is integrated with the A-Frame Registry. From the Inspector, you can install components from the Registry and attach them to objects in the scene with a couple of clicks.

4.2.5 Device and Platform Support

Devices and platform support depends on how well the browsers support certain devices and APIs. A-Frame supports both flat (3D on a normal screen) and WebVR experiences, though its focus is heavily VR.



ID	torusKnot88		
position	1.13	8.03	-12.22
rotation	48.13	0.00	-125.48
scale	1.00	1.00	1.00
visible	✓		
mixins	light		
Add	blend-character-model		
buffer	✓		
mergeTo			
primitive	torusKnot		
skipCache	☐		
p	8.00		
q	7.00		
radius	9.49		
radiusTubular	0.25		
segmentsRadial	8		
segmentsTubular	100		

Figure 4 – 1 A-Frame Inspector window.

Support for flat experiences primarily depends on a browser's WebGL support. We can see which browsers support WebGL at http://caniuse.com/feat=webgl.

Support for VR experiences, along with WebGL support, depends on a browser's support for the WebVR API. You can see which browsers currently support the WebVR API at Is WebVR Ready?. A-Frame supports version 1.0 of the WebVR API.

The exception to this is for mobile devices, which are supported through the WebVR Polyfill.

A-Frame currently supports 6DoF controllers using Brandon Jones' 4/24/16 experimental build of Chromium with experimental controller support (e.g., Vive controllers).

4.2.6 Hardware specification

To use Vive, your computer must meet the following minimum system requirements.

- GPU: NVIDIA® GeForce® GTX 970, AMD Radeon^{TM} R9 290 equivalent or better
- CPU: Intel® Core^{TM} i5-4590/AMD FX^{TM} 8350 equivalent or better
- RAM: 4 GB or more
- Video output: HDMI 1.4, DisplayPort 1.2 or newer
- USB port: 1x USB 2.0 or better port
- Operating system: Windows® 7 SP1, Windows® 8.1 or later, Windows® 10

4.3 React

ReactJS is a javascript library for building user interfaces, originally created by engineers at Facebook to solve the challenges involved when developing complex user interfaces with datasets that change over time. It provides a way to write encapsulated components that manage their own state, then compose them to make complex user interfaces. It doesn't make assumptions about the rest of the technology stack, because it's just library. Since component logic is written in JavaScript instead of templates, we can easily pass rich data through our app and keep state out of the DOM.

4.4 Flux

Flux is an application architecture created by Facebook for building client-side web applications. It complements the React in a way that displaces the standard Model-View-Controller (MVC) framework. It was designed in React in mind and aims to help developers to create more efficient, maintainable code when dissecting the application into multiple components, which according to Facebook engineers was hard if the traditional MVC pattern was followed when building the application with React. (Gackenheimer, 2015)

Flux is utilizing a unidirectional data flow. It's more of a pattern rather than a formal framework, and developers can start using Flux immediately without a lot of new code.

Flux applications have three major parts: the dispatcher, the stores, and the views (React components). These should not be confused with Model-View-Controller. Controllers do exist in a Flux application, but they are controller-views - views often found at the top of the hierarchy that retrieve data from the stores and pass this data down to their children. Additionally, action creators - dispatcher helper methods - are used to support a semantic API that describes all changes that are possible in the application.

Flux abandons MVC in favor of a uni-directional data flow. When a user interacts with a React view, the view propagates an action through a central dispatcher, to the various stores that hold the application's data and business logic, which updates all of the views that are affected. This works especially well with React's declarative programming style, which allows the store to send updates without specifying how to transition views between states. (Facebook, Inc., 2015)

4.4.1 Structure and Data Flow in Flux architecture

A uni-directional data flow is the main consequence of using the Flux pattern, and the diagram below should be the primary mental model for the Flux programmer. The dispatcher, stores and views are independent nodes with distinct inputs and outputs. The actions are simple objects containing a action type that identifies the property and the new data that is passed along with the action. (Facebook, Inc., 2015)

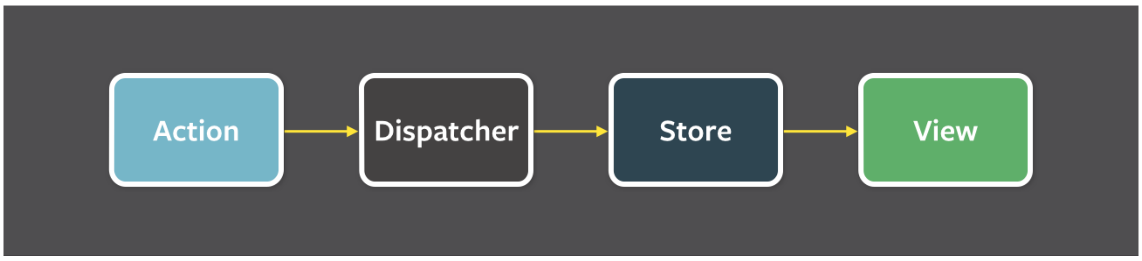


Figure 4 – 2 Data in a Flux application flows in a single direction.

The views may cause a new action to be propagated through the system in response to user interactions.

All data flows through the dispatcher as a central hub. Actions are provided to the dispatcher in an action creator method, and most often originate from user interactions with the views. The dispatcher then invokes the callbacks that the stores have registered with it, dispatching actions to all stores. Within their registered callbacks, stores respond to whichever actions are relevant to the state they maintain. The stores then emit a change event to alert the controller-views that a change to the data layer has occurred. Controller-views listen for these events and retrieve data from the stores in an event handler. The controller-views call their own setState() method, causing a re-rendering of themselves and all of their descendants in the component tree.

This structure allows us to reason easily about our application in a way that is reminiscent of functional reactive programming, or more specifically data-flow programming or flow-based programming, where data flows through the application in a single direction — there are no two-way bindings. Application state is maintained only in the stores, allowing the different parts of the application to remain highly decoupled.

We found that two-way data bindings led to cascading updates, where changing one object led to another object changing, which could also trigger more updates. As applications grew, these cascading updates made it very difficult to predict what would

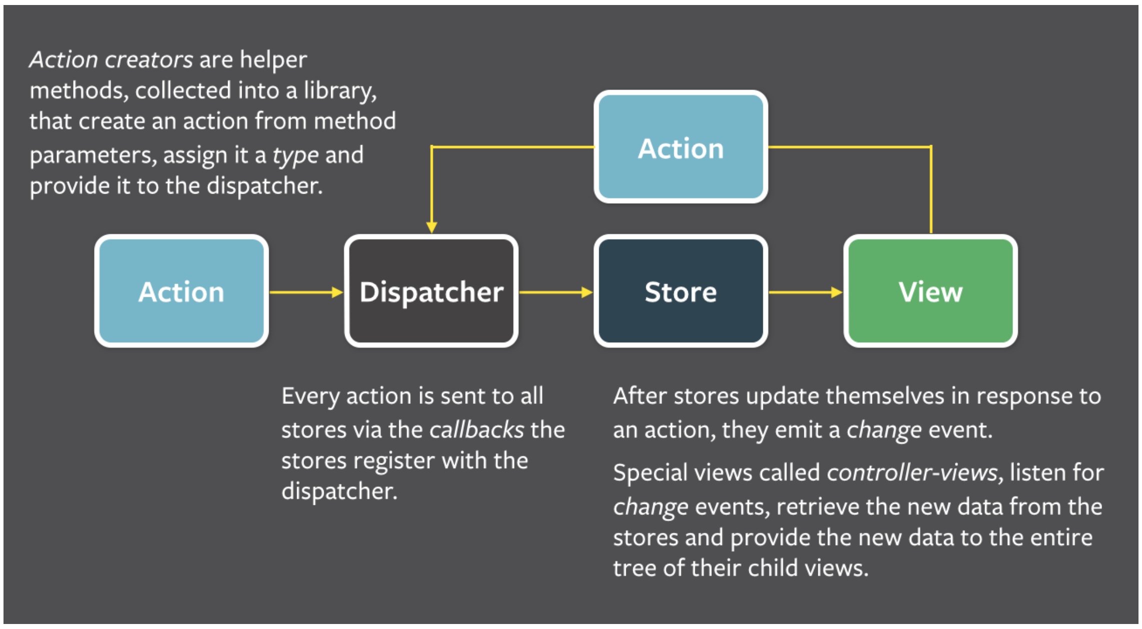

Action creators are helper methods, collected into a library, that create an action from method parameters, assign it a *type* and provide it to the dispatcher.
Action
Action
Dispatcher
Store
View
Every action is sent to all stores via the *callbacks* the stores register with the dispatcher.
After stores update themselves in response to an action, they emit a *change* event.
Special views called *controller-views*, listen for *change* events, retrieve the new data from the stores and provide the new data to the entire tree of their child views.

Figure 4 – 3 Detailed Flux application architecture diagram.

change as the result of one user interaction. When updates can only change data within a single round, the system as a whole becomes more predictable.(Facebook, Inc., 2015)

4.5 Redux

Redux is a predictable state container for JavaScript apps. It helps us write applications that behave consistently, run in different environments (client, server, and native), and are easy to test. On top of that, it provides a great developer experience, such as live code editing combined with a time traveling debugger. We are using Redux together with React, but in general it can be used with any other view library.

The whole state of application is stored in an object tree inside a single store. The only way to change the state tree is to emit an action, an object describing what happened. To specify how the actions transform the state tree, we write pure reducers.

Instead of mutating the state directly, we specify the mutations we want to happen with plain objects called actions. Then we write a special function called a reducer to decide how every action transforms the entire application's state.

The beauty and strength of this pattern is how well it scales to large and complex apps. It also enables very powerful developer tools, because it is possible to trace every mutation to the action that caused it. We can record user sessions and reproduce them just by replaying every action.

4.6 Webpack

Webpack is a module bundler for modern JavaScript applications. It takes modules with dependencies and generates static assets representing those modules [4−4]. Its strength is that it's configurable and developers using it in their applications should know the four core concepts used when configuring Webpack:

- Entry - Webpack creates a graph of all of your application's dependencies and the starting point of this graph is called entry point. It tells Webpack where to start and follows the graph of dependencies to know what to bundle.
- Output - it tells Webpack where to bundle our application.
- Loaders - Webpack treats every file (.css, .html, .scss, .jpg, etc.) as a module. However, webpack only understands JavaScript, so loaders transform these files into modules as they are added to dependency graph.
- Plugins - loaders only execute transform on per-file basis, but plugins are mostly used to perform actions and custom functionality on combilations or chunks of bundled modules.

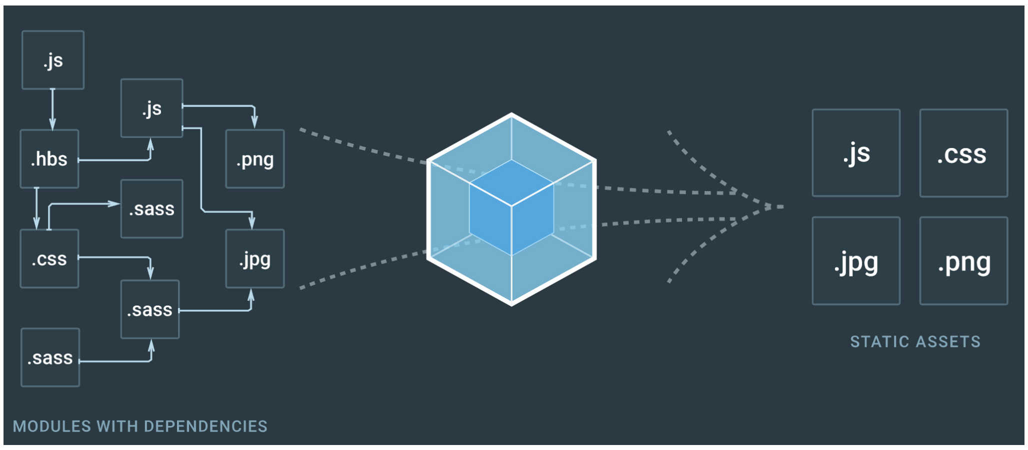


Figure 4 – 4 Bundling dependencies and static assets with Webpack

5 Conceptualisation and design

5.1 Inputs and objections

5.2 Functionality planning

5.3 Virtual user interface

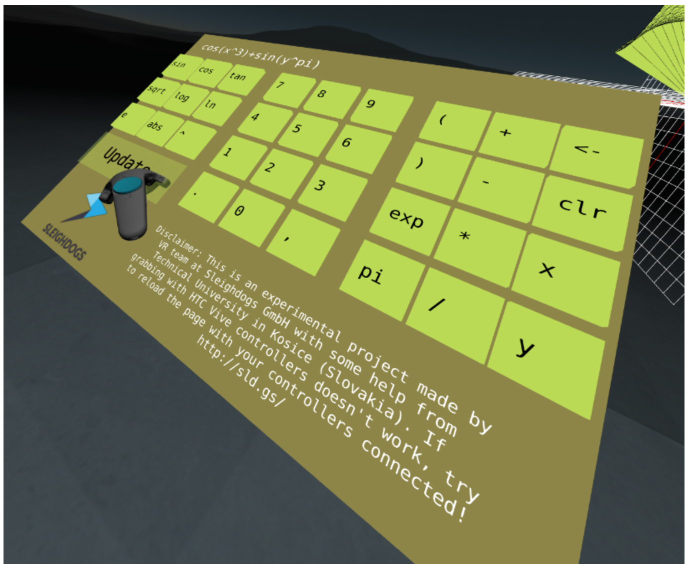
6 Implementation of virtual reality application

6.1 Math.js

Math.js is an extensive math library for JavaScript and Node.js. It features a flexible expression parser with support for symbolic computation, comes with a large set of built-in functions and constants, and offers an integrated solution to work with different data types like numbers, big numbers, complex numbers, fractions, units, and matrices.

6.2 Test-driven development

Test-driven development (TDD), is an evolutionary approach to development which combines test-first development where we write a test before we write just enough production code to fulfill that test and refactoring. One view is the goal of TDD is specification and not validation. In other words, it's one way to think through our requirements or design before we write our functional code (implying that TDD is both an important agile requirements and agile design technique). Another view is that TDD is a programming technique. The goal of TDD is to write clean code that works.



Disclaimer: This is an experimental project made by VR team at Sleighdogs GmbH with some help from Technical University in Kosice (Slovakia). If grabbing with HTC Vive controllers doesn't work, try to reload the page with your controllers connected! http://sld.gs/

Figure 6 – 1 Virtual calculator entity 6 – 1

6.3 Virtual calculator entity

6.3.1 Parsing equations with Math.js

6.4 Parametrized function grid entity

6.5 Interactive function settings panel entity

6.6 Interactions

6.6.1 Grabbing

6.6.2 Scaling

6.7 Building and deploying the application to web hosting

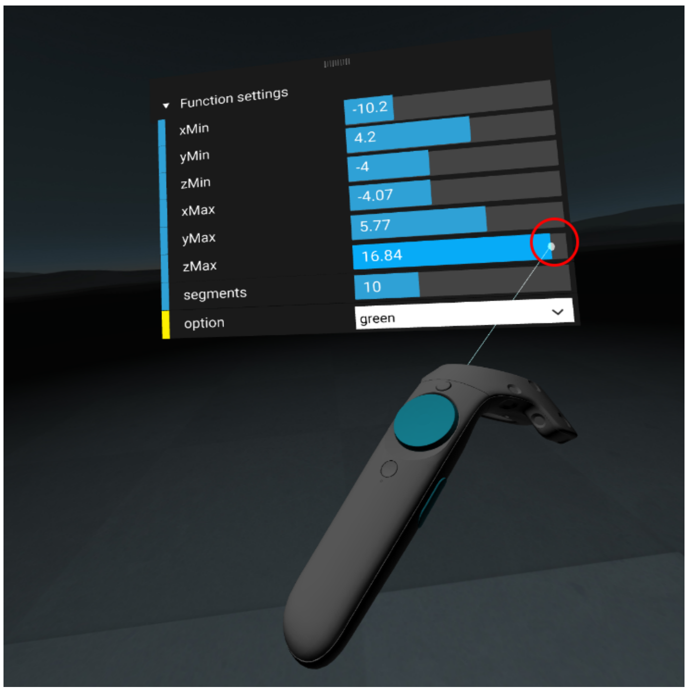


Figure 6 – 2 Function settings entity 6 – 2

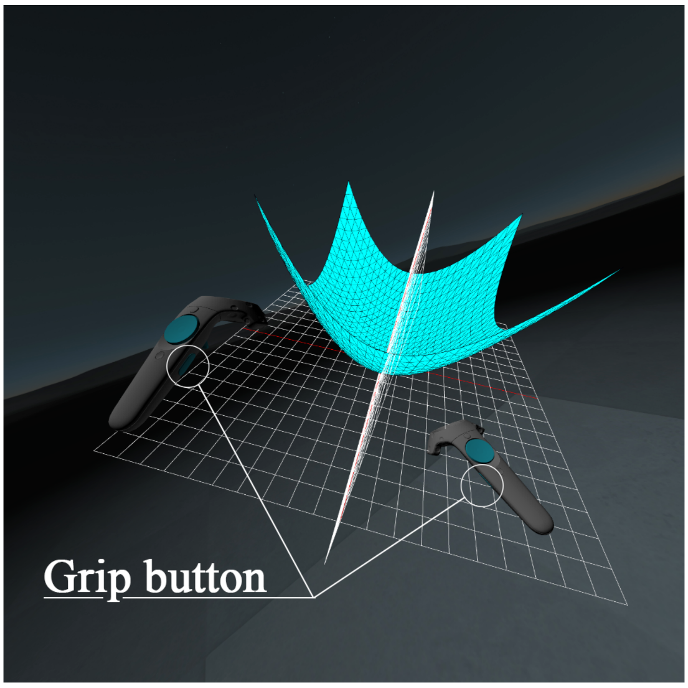


Figure 6 – 3 Grabbing functionality 6 – 3

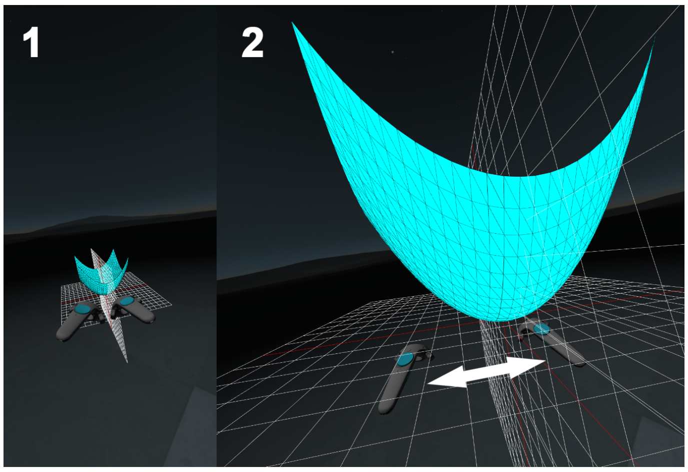


Figure 6 – 4 Scaling functionality 6 – 4

7 Conclusion

Táto časť záverečnej práce je povinná. Autor uvedie zhodnotenie riešenia. Uvedie výhody, nevýhody riešenia, použitie výsledkov, ďalšie možnosti a pod., prípadne načrtne iný spôsob riešenia úloh, resp. uvedie, prečo postupoval uvedeným spôsobom.

References

LOEFFLER, C. 1995. *Distributed Virtual Reality: Applications for Education, Entertainment and Industry.* http://www.nta.no/telektronikk/4.93.dir/Loeffler_C_E.html

KAUFMANN, H. 2011. *Virtual Environments for Mathematics and Geometry Education.* In: Themes In Science and Technology Education. Vienna : Klidarithmos Computer Books, 2011, Special Issue, pp. K 131–152

MOZILLA, 2017. *A-Frame Introduction.* https://aframe.io/docs/0.5.0/introduction/

GACKENHEIMER, C. 2015. *Introduction to React.* USA : Apress, 2015 129 s. ISBN 978-1-4842-1245-5

FACEBOOK, Inc. 2015. *Flux - In Depth Overview.* https://facebook.github.io/flux/docs/in-depth-overview.html