**Stress Management through AR VR**

**By**

**Karan Mungra (19BCE137)**

**Nikhil Makwana(19BCE107)**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**Ahmedabad 382481**

**Stress Management through AR VR**

**Report**

Submitted in partial fulfilment of the requirements

For the degree of

**Bachelor of Technology in Computer Science & Engineering**

By

**Karan Mungra (19BCE137) ,Nikhil Makwana(19BCE107)**

Guided By

**Dr. Mohd Zuhair**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**Ahmedabad 382481**

**CERTIFICATE**

This is to certify that the Minor Project entitled “Stress Management through AR VR” submitted by **Karna Mungra (19BCE137) ,Nikhil Makwana(19BCE107)** towards the partial fulfillment of the requirements for the degree of Bachelor of Technology inComputer Science and Engineering of Nirma University is the record of work carried out by him/her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination.

**Dr. Mohd zuhair** **Dr. Madhuri Bhavsar**

Assistant Professor, Professor and HOD

Computer Science and Engineering Dept Computer Science and Engineering dept

Institute of Technology, Institute of Technology,

Nirma University. Nirma University.

**Acknowledgement**

I have taken efforts in this project work. However, it would not have been possible without the kind support and help of many individuals. I would like to extend my sincere thanks to all of them. I am highly indebted to Dr. Mohd Zuhair for her guidance and constant supervision as well as for providing necessary information regarding the project work. I would like to express my gratitude towards my parents and member of family for their kind co-operation and encouragement which help me in completion of this project. My thanks and appreciations also go to people who have willingly helped me out with their abilities.

Karan Mungra & Nikhil Makwana

#### ABSTRACT

#### This report provides description and methodology of the AR/VR THREEJS model for terrain rendering. The model is also applied to use case of stress management using AR/VR Natural Forest Rendering. The model is designed to be flexible, scalable and easily enhanced to meet the requirements. It provides features like View rendering relative camera focus (which enables) to provide illusion of infinite terrain, basic building blocks for rendering forest in the model like Ground, Grass Objects, Tree Objects, etc. It builds upon THREEJS framework, WEBGL Engine and WebXR API.

The main objective of the model is providing a basic framework to be used for building better AR/VR nature rendering projects for stress management. A comprehensive list of framework components is RelativeViewWorld, RelativeCamera, TerrainModel, TreeModel, RockModel, HillModel, BushesModel, etc.

#### CONTENTS

# Certificate I

**Acknowledgement** Ii

**Abstract** Ii

**Chapter 1 Introduction**

**Chapter 2 Literature overview**

**Chapter 3 Methodology**

**Chapter 4 Results**

**Chapter F Summary conclusion and Future work**

**Reference**

**Introduction**

Stress management means to control stress and reduce the stress gradually through a period through a habit of activities. One of such activities in current time has come to be VR/AR technology. The technology is able to provide an virtual rendered world and also capable the user to interact with the world. In a recent example of this technology used in real life is of US Airforce, which has created an virtual funeral ceremony for Iraq War, so the veterans can re-experience the funeral of their lost comrades. The activity can reduce stress and anxiety of the war by relieving the experience of pain. And enhance the veterans to deal with the trauma.

Experience of the natural world can also reduce stress overtime in particular cases. But not always possible for people dealing with diseases or disabilities to travel to such places. Also in this fast-pacing world, for people to deal with their daily works and at the same time to go a mini vacation regularly is not possible. Hence, an VR/AR model can give similar experience to the user. Hence, we have developed a basic modular framework or a collection of tools to easily developed such worlds using THREEJS and WEBGL.

The framework is created to provide basic elements necessary for such virtual world rendering. It also provides enough flexibility, that a user can quickly come up with a virtual world.

It does not provide any highly complex tool which is not easily modifiable. One can always make tweaks in the original source code according to use case and can get better performance. Still, the model can easily deal with advanced usage for rendering. It provides features like View rendering relative to camera for better performance.

The main objective of the framework is to provide flexibility, simplicity in usage and performance.

**Literature overview**

[1] Bhargava, Deepti & Trivedi, Hemant. (2018). A Study of Causes of Stress and Stress Management among Youth. IRA-International Journal of Management & Social Sciences (ISSN 2455-2267). 11. 108. 10.21013/jmss.v11.n3.p1.

The paper discusses about causes and effects of stress in youth and provides how to combat with stress at period of life from childhood to adolescence. It gives various suggestions such as positive environment, creating hostile learning environment by minimizing negative impact of stressors for stress management. The paper also emphasizes on the success of development stage of child into adolescence.

[2] Alborzkouh P, Nabati M, Zainali M, Abed Y, Shahgholy Ghahfarokhi F. A review of the effectiveness of stress management skills training on academic vitality and psychological well-being of college students. J Med Life. 2015;8(Spec Iss 4):39-44. PMID: 28316704; PMCID: PMC5319270.

The paper reviews the effectiveness of stress management training on the academic life and mental well-being of the students of Shaded University. The researchers sample out a total of 40 students through convenience sampling method and were organized into two groups: experimental and control group. Then both group were pretested by using an academic vitality inventory and an 84 – questions psychological well-being inventory. Then, the experimental group received stress management skills training for ten sessions, and the control group did not receive any intervention. Next, both groups were post-tested, and the data were analysed with SPSS-21 software by using descriptive and inferential statistical methods. The findings showed that the stress management skills training significantly contributed to promoting the academic vitality and psychological well-being of students (p < 0.001). Hence, from the paper one can conclude through proper stress management activities can lead stress reduction.

[3] AUTHOR=Kim Hyewon, Kim Dong Jun, Kim Seonwoo, Chung Won Ho, Park Kyung-Ah, Kim James D. K., Kim Dowan, Kim Min Ji, Kim Kiwon, Jeon Hong Jin

TITLE=Effect of Virtual Reality on Stress Reduction and Change of Physiological Parameters Including Heart Rate Variability in People With High Stress: An Open Randomized Crossover Trial

JOURNAL=Frontiers in Psychiatry

VOLUME=12YEAR=2021URL=https://www.frontiersin.org/articles/10.3389/fpsyt.2021.614539 DOI=10.3389/fpsyt.2021.614539 ISSN=1664-0640

This paper focuses on measuring the effectiveness of VR technology in stress management. The researchers tested the stress management through VR on a group of participants. Participants consisted of 83 healthy adult volunteers with high stress, which was defined as a score of 20 or more on the Perceived Stress Scale-10 (PSS-10). This study used an open, randomized, crossover design with baseline, stress, and relaxation phases. During the stress phase, participants experienced an intentionally generated shaking VR and serial-7 subtraction. For the relaxation phase, participants underwent a randomly assigned relaxation session on day 1 among VR relaxation and biofeedack, and the other type of relaxation session was applied on day 2. We compared the State-Trait Anxiety Inventory-X1 (STAI-X1), STAI-X2, the Numeric Rating Scale (NRS), and physiological parameters including heart rate variability (HRV) indexes in the stress and relaxation phases.

**Results [referenced from the paper]:** A total of 74 participants were included in the analyses. The median age of participants was 39 years, STAI-X1 was 47.27 (SD = 9.92), and NRS was 55.51 (SD = 24.48) at baseline. VR and biofeedback significantly decreased STAI-X1 and NRS from the stress phase to the relaxation phase, while the difference of effect between VR and biofeedback was not significant. However, there was a significant difference in electromyography, LF/HF ratio, LF total, and NN50 between VR relaxation and biofeedback.

In conclusion, they found VR technology is quite effective for stress management.

**Methodology**

**Technologies Used:**

* WEBGL 3

WebGL (Web Graphics Library) is a JavaScript API for rendering high-performance interactive 3D and 2D graphics within any compatible web browser without the use of plug-ins. WebGL does so by introducing an API that closely conforms to OpenGL ES 2.0 that can be used in HTML <canvas> elements. This conformance makes it possible for the API to take advantage of hardware graphics acceleration provided by the user's device.

* THREEJS

Three.js is often confused with WebGL since more often than not, but not always, three.js uses WebGL to draw 3D. WebGL is a very low-level system that only draws points, lines, and triangles. To do anything useful with WebGL generally requires quite a bit of code and that is where three.js comes in. It handles stuff like scenes, lights, shadows, materials, textures, 3d math, all things that you'd have to write yourself if you were to use WebGL directly.

* WEBXR API

WebXR is a group of standards which are used together to support rendering 3D scenes to hardware designed for presenting virtual worlds (virtual reality, or VR), or for adding graphical imagery to the real world, (augmented reality, or AR). The WebXR Device API implements the core of the WebXR feature set, managing the selection of output devices, render the 3D scene to the chosen device at the appropriate frame rate, and manage motion vectors created using input controllers.

**Understanding Webgl:**

Webgl is a rasterization engine. And it runs on the GPU. The whole development of the project depends on how the WebGl works. Webgl provides very minimal of things that can be done using less code. It only provides three different types of primitives or basic shapes that it can rasterize: Points, Lines and Triangles.

It works using a shader program. Whose job it is to run on the GPU and draw the shape according to provided data. You provide that code in the form of pairs of functions. Those 2 functions are called a vertex shader and a fragment shader and they are each written in a very strictly typed C/C++ like language called GLSL. (GL Shader Language). Paired together they are called a program.

A vertex shader's job is to compute vertex positions. Based on the positions the function outputs WebGL can then rasterize various kinds of primitives including points, lines, or triangles. When rasterizing these primitives it calls a second user supplied function called a fragment shader. A fragment shader's job is to compute a color for each pixel of the primitive currently being drawn.

Nearly all of the entire WebGL API is about setting up state for these pairs of functions to run. For each thing you want to draw you setup a bunch of state then execute a pair of functions by calling gl.drawArrays or gl.drawElements which executes your shaders on the GPU.

Any data you want those functions to have access to must be provided to the GPU. There are 4 ways a shader can receive data.

Buffers are arrays of binary data you upload to the GPU. Usually buffers contain things like positions, normals, texture coordinates, vertex colors, etc although you're free to put anything you want in them.

Attributes are used to specify how to pull data out of your buffers and provide them to your vertex shader. For example you might put positions in a buffer as three 32bit floats per position. You would tell a particular attribute which buffer to pull the positions out of, what type of data it should pull out (3 component 32 bit floating point numbers), what offset in the buffer the positions start, and how many bytes to get from one position to the next.

Buffers are not random access. Instead a vertex shader is executed a specified number of times. Each time it's executed the next value from each specified buffer is pulled out and assigned to an attribute.

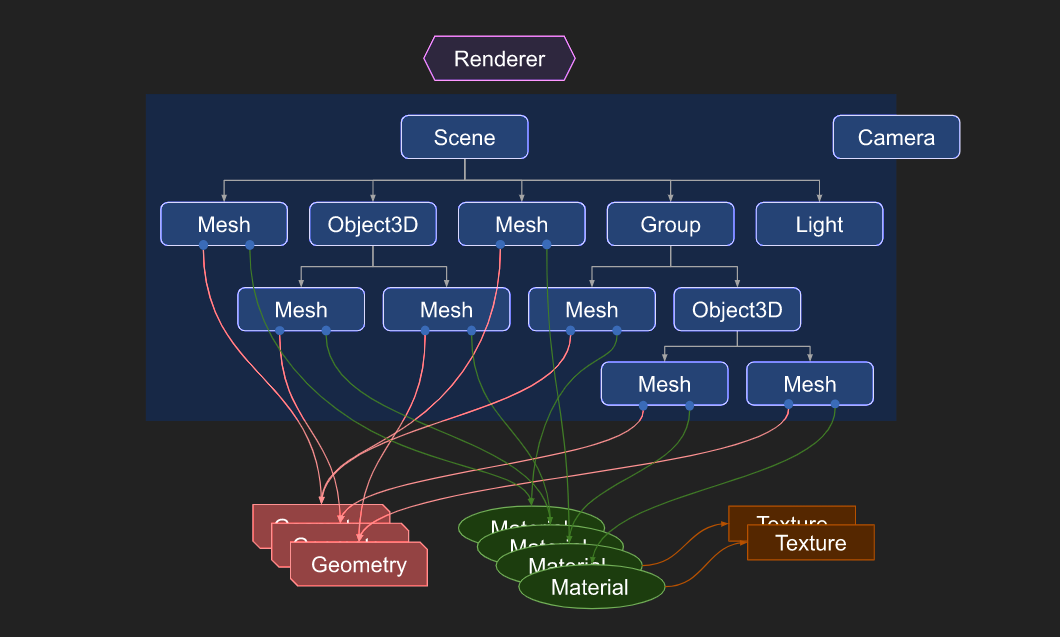
Uniforms are effectively global variables you set before you execute your shader program.

Textures are arrays of data you can randomly access in your shader program. The most common thing to put in a texture is image data but textures are just data and can just as easily contain something other than colors.

Varyings are a way for a vertex shader to pass data to a fragment shader. Depending on what is being rendered, points, lines, or triangles, the values set on a varying by a vertex shader will be interpolated while executing the fragment shader.

**How to create a world in Threejs:**

Three.js is often confused with WebGL since more often than not, but not always, three.js uses WebGL to draw 3D. WebGL is a very low-level system that only draws points, lines, and triangles. To do anything useful with WebGL generally requires quite a bit of code and that is where three.js comes in. It handles stuff like scenes, lights, shadows, materials, textures, 3d math, all things that you'd have to write yourself if you were to use WebGL directly.



*Figure:* [*https://threejs.org/manual/#en/fundamentals*](https://threejs.org/manual/#en/fundamentals)

Threejs have mainly three main components a renderer whose task is to render onto the screenView, a scene which contains all the objects which are going to be visible in the view and a camera which creates the projectionMatrix.

There is a Renderer. This is arguably the main object of three.js. You pass a Scene and a Camera to a Renderer and it renders (draws) the portion of the 3D scene that is inside the frustum of the camera as a 2D image to a canvas.

There is a scenegraph which is a tree like structure, consisting of various objects like a Scene object, multiple Mesh objects, Light objects, Group, Object3D, and Camera objects. A Scene object defines the root of the scenegraph and contains properties like the background color and fog. These objects define a hierarchical parent/child tree like structure and represent where objects appear and how they are oriented. Children are positioned and oriented relative to their parent. For example the wheels on a car might be children of the car so that moving and orienting the car's object automatically moves the wheels. You can read more about this in the article on scenegraphs.

Note in the diagram Camera is half in half out of the scenegraph. This is to represent that in three.js, unlike the other objects, a Camera does not have to be in the scenegraph to function. Just like other objects, a Camera, as a child of some other object, will move and orient relative to its parent object. There is an example of putting multiple Camera objects in a scenegraph at the end of the article on scenegraphs.

Mesh objects represent drawing a specific Geometry with a specific Material. Both Material objects and Geometry objects can be used by multiple Mesh objects. For example to draw two blue cubes in different locations we could need two Mesh objects to represent the position and orientation of each cube. We would only need one Geometry to hold the vertex data for a cube and we would only need one Material to specify the color blue. Both Mesh objects could reference the same Geometry object and the same Material object.

Geometry objects represent the vertex data of some piece of geometry like a sphere, cube, plane, dog, cat, human, tree, building, etc... Three.js provides many kinds of built in geometry primitives. You can also create custom geometry as well as load geometry from files.

Material objects represent the surface properties used to draw geometry including things like the color to use and how shiny it is. A Material can also reference one or more Texture objects which can be used, for example, to wrap an image onto the surface of a geometry.

Texture objects generally represent images either loaded from image files, generated from a canvas or rendered from another scene.

Light objects represent different kinds of lights.

To create a world, all we need to do is to setup the right camera positions, lights and place objects in the scene. That is all required.

**Conclusion**

Many things are possible in this range of technology. It all depends on creativity and performance of the rendering devices. One can create a whole 3d forest experience and create stress reducing tool for a specified group of people.

**References**

Carson, Daniel. (2020). Research-Paper-WebGL. 10.13140/RG.2.2.18538.95683.

https://threejs.org/docs/#api/en/materials/MeshPhongMaterial