RESEARCH ARTICLE OPEN ACCESS

Learning Physics And Biology In Virtual Laboratory

Anand Deshpande*, Deval Shah**, Rohit Kawale***, Saif Vazir****
**(Software Engineer, Samsung R&D Institute, Bangalore, India*
***(Jamnalal Bajaj Institute of Management Studies (JBIMS), Mumbai, India*
****(Software Engineer, Barclays, Pune, India*
*****(Department of Computer Science, Stony Brook University, NY, USA*
Corresponding Author: Anand Deshpande

ABSTRACT: It is very common to find educational institutions in rural India where there are no equipments available for performing academic experiments. This problem can be mitigated with the help of Virtual Reality. We have used Google Cardboard as the Head Mounted Display and Leap Motion for interacting with the application. The cost of these devices is low so, we have succeeded in creating a virtual laboratory at a cost which is less than the cost of setting up and maintaining a real laboratory. Although it has been developed at low cost, it has a few drawbacks. Using the application requires large amount of practice and prolonged usage can lead to nausea and disorientation. VR is still in the early stages, but it has huge potential. This application can be further expanded to include a wide range of laboratories like Chemistry and Mechanics. If this low cost solution can be implemented in schools and colleges, it will provide facilities to students and save a huge amount of money for institutions. Effective implementation of such a solution will improve the quality of practical education to a great extent. With further improvements in VR hardware, the problems of accuracy, portability and VR sickness can be solved.

Keywords: Cardboard, LeapMotion, Unity, Virtual Reality, Physics Laboratory, Biology Laboratory

---

Date Of Submission:04-09-2018 Date Of Acceptance: 20-09-2018

I. INTRODUCTION

Large number of students, in developing countries like India, do not have the access to adequate resources in the practical laboratories of their educational institutions. The outcome of this is, even though the students have the theoretical knowledge required, they lack the understanding associated with the practical implementation of the theory. This leads to low quality of the workforce. The major reason for this problem is the lack of monetary reserves in the institutions. The purpose of the project is to create a laboratory using Virtual Reality[1] with the use of minimum hardware, so that the institutions can provide better opportunities to the students at lower costs. The scope of our project involves determining how effectively the virtual laboratory can be developed and the cost of development. If the cost of the hardware for the virtual laboratory is higher than the physical equipments required for the laboratory, then such a virtual laboratory would be of no use. However, due to the progress of computer technology, the cost of various hardware devices has been reducing and it would not be difficult to find an economic device suitable for our use. Apart from the purchase of hardware, there is no other major expense as there are various softwares available which can be used for the development of the project. Virtual laboratories have the capability to help the students improve their practical knowledge in the absence of well-equipped physical laboratories. All the problems that occur in the physical laboratories due to the absence of ideal conditions in our real world, can also be easily eliminated by creating ideal conditions in the virtual laboratory. Even the human errors can be reduced with the help of efficient hardware devices. In the real world laboratories, the maintenance and upgradation of the equipment requires large amount of money as well as skilled professionals. This increases the burden on the educational institutions. On the other hand, there is no cost associated with the maintenance of the virtual laboratory. Also, the upgradation can done be easily with the release of a new version of the software. There is no existing system which implements Virtual Laboratories. This system is a novel idea and will be very helpful for students as well as colleges. Creating a virtual laboratory at a cost which is less than the cost of setting up and maintaining it, could revolutionize the education sector. This virtual laboratory system is aimed at providing practical education which is open to all students.

II. PROJECT SETUP

2.1 Devices Used

2.1.1 Google Cardboard Head Mount Display (HMD)

Google Cardboard[2] headsets are built out of simple, low-cost components. On assembling the headset, a smartphone is inserted in the back of the device and held in place by the selected fastening device. A Google Cardboard–compatible app will split the smartphone display image into two halves. This results in a stereoscopic (”3D”) image with a wide field of view. The cost advantage provided by Google Cardboard is unparalleled and hence is the option chosen by us for our project. The availability of Cardboard based viewers supports our proposition for a readily available education system which is open for all.

2.1.2 Leap Motion

The Leap Motion[3] controller is a small USB peripheral device which is used to track hand gestures. It is designed in such a way that it can be placed on a physical desktop, facing upward. It can also be mounted onto a virtual reality headset. It contains two monochromatic IR cameras and three infrared LEDs, which track hand gestures to a distance of about 1 meter in a hemispherical area. The Leap Motion controller is available at a price that is far less as compared to other controller devices which include HTC Vive, Oculus Rift and Samsung Gear VR. Apart from its cost benefit, the precise and accurate gesture tracking offered by the device has been the reason for its selection for our project. Also Leap Motion can be used along with Unity by using the Unity Core Assets for Leap Motion.

2.2 Software Used

2.2.1 Unity Game Engine

Unity is a game development engine which supports the development of Virtual Reality using C sharp. Unity provides a user friendly graphical user interface (GUI) which helps the developers to modify the VR Scenes without much hassle. Due to the vast expanse of platforms and devices supported, Unity was our go to development engine. For our project, we have used the Unity Asset Store to fetch some of the pre-made environments and objects which reduce the developing load and promote the fundamentals of Code Reuse. We are using Unity Engine version 5.3+ personal edition.

2.2.2 Blender

Blender[4] is the free and open source 3D creation suite. It supports the entirety of the 3D pipeline. This includes modeling, rendering, compositing as well as animation, simulation and motion tracking. In addition it also supports video editing and game creation. The open source community, along with a number of tutorials, readily available plugins for modelling, have made Blender an ideal choice for creating models of different environments and objects which are difficult to execute in Unity Engine.

III. IMPLEMENTATION

3.1 Virtual Experiments

Table 3.1: Physics Experiments

Physics Experiments
1 Acceleration due to gravity To calculate the acceleration due to gravity (g) using a simple pendulum in virtual environment. The following formula has been used- $T = 2\pi\sqrt{\frac{L}{g}}$
2 Spring constant To calculate the spring constant (k) of a spring in virtual environment. The following formula has been used- $T = 2\pi\sqrt{\frac{m}{k}}$
3 Vernier Calliper To calculate the length using Vernier Calliper in virtual environment.
4 Conduction of electricity To show, in virtual environment, that distilled water is a bad conductor of electricity while salt water is a good conductor of electricity.
5 Rolling bodies To calculate acceleration of rolling bodies and verify with the theoretical values in virtual environment. The following formulae have been used- To calculate theoretical value: $a = \frac{g\sin(\theta)}{\left(I + mr^2\right)}$ To calculate practical value: $s = \frac{1}{2}at^2$

Table 3.2: Biology Experiments

Biology Experiments
1 Heart To study the anatomy of human heart in virtual environment. The following parts have been studied- Heart Chambers Inferior Vena Cava Superior Vena Cava Pulmonary Artery Pulmonary Vein
2 Neuron To study the anatomy of neuron in virtual environment. The following parts have been studied- Axon Terminal Dendrite Myelin Sheath Cell Body
3 Skeleton To study the anatomy of human skeleton in virtual environment. The following parts have been studied- Skull Rig cage Hands Legs Spine
4 Brain To study the anatomy of human brain in virtual environment. The following parts have been studied- Frontal Lobe Occipital Lobe Parietal Lobe Temporal Lobe Cerebellum Brainstem
5 Dinosaur To study about Allosaurus in virtual environment.

3.2 Development Methodology

The block diagram presented in Fig 3.1 describes the development process of the application. The process starts with collecting the premade assets. These assets can be collected from the Unity asset store. Some assets can also be imported from the Blender market. After asset collection, if some assets cannot be found on the asset store, we can model those assets using Blender software. After model creation, we create the layout of the entire scene. After the scene has been created, we include the object-scene interactivity. This process is completed by creating C sharp scripts which enable the objects to interact with the scenes and also perform various predefined animations on certain inputs. We then move onto the final process of incorporating user inputs and enabling the user to interact with the virtual environment created. This is done using VRInput method defined in Unity library. Inputs are obtained from the gyroscope readings of mobile as well as leap motion device. This concludes the development process of the application.

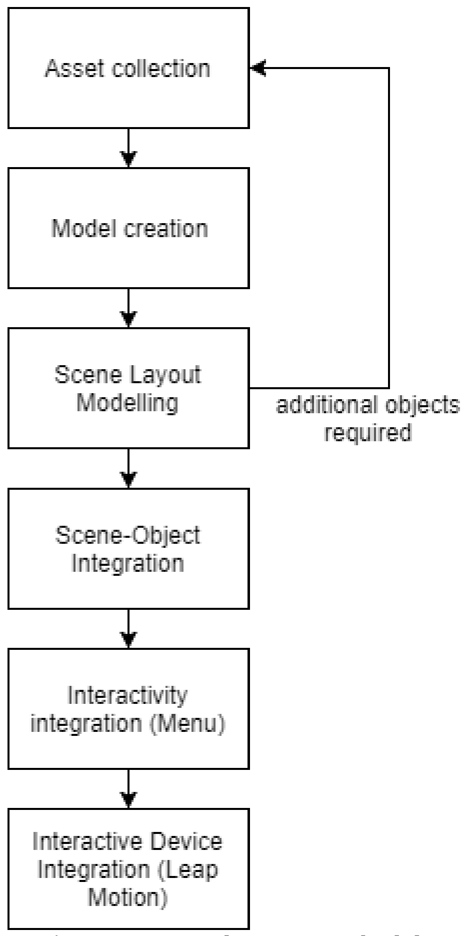

Asset collection
Model creation
Scene Layout Modelling
additional objects required
Scene-Object Integration
Interactivity integration (Menu)
Interactive Device Integration (Leap Motion)

Figure 3.1: Development Methodology

3.3 Stereoscopic Rendering

Stereoscopy[5] is a method by which the illusion of depth in an image is created or enhanced by means of stereopsis for binocular vision. Images that appear three-dimensional when wearing special glasses can be generated using stereo cameras. This effect can be achieved by rendering two separate images from cameras that are a small distance apart from each other, similar to the mechanism used by our eyes. When viewing a stereo image, the left eye is limited to seeing one of the images, while the right eye sees the second image. Our brain then merges these two images together, making it appear as if we are looking at a 3D object ratherthan a flat image. In stereoscopic rendering the device screen is split in two parts generating views for left and right eye separately this gives the effect of 3D visualisation of the virtual environment. We use stereoscopic rendering for viewing the lab in 3D on our head mounted display (Google Cardboard).

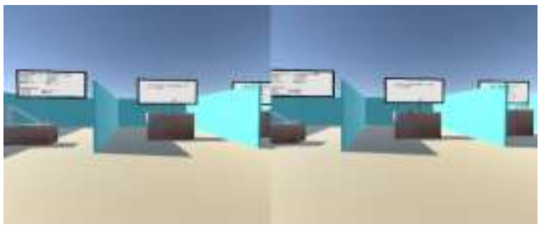
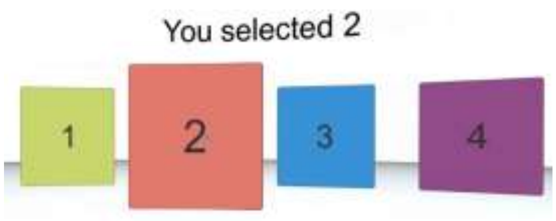


Figure 3.2: Stereoscopic View of Virtual Laboratory

3.4 Gaze Interaction

In VR, there might be a need to activate an object that a user is looking at. This is accomplished using Gaze Interaction. A script is written, which casts a ray forwards to see if the ray hits any colliders. When a collider is hit by a ray, the script finds the interactive component on the object. So, we can determine whether the user is looking at an object or not, and call the appropriate method.



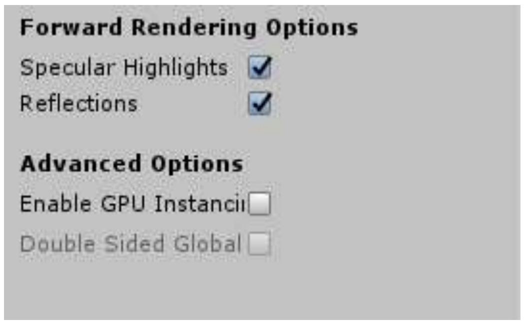
Source:https://productcoalition.com/ui-interaction-in-mobile-virtual-reality-4-reasons-to-not-use-the-touch-pad-on-samsungs-gear-vr-6b1c86a8f140
Figure 3.3: Gaze Interaction

3.5 GPU Instancing

GPU Instancing, also known as Geometry instancing[6][7], is the process of rendering multiple copies of the same mesh which are present in a scene, at once. It uses only a small number of draw calls[8] as opposed to standard instancing. This helps in significantly increasing the performance of the application[9]. This process can be used only for objects which have identical Meshes but can vary in terms of their dimension and colour. To understand the benefits of GPU Instancing, we need to understand how an object is drawn on the screen. To draw an object, an API call is made to the graphics driver(such as OpenGL or Direct3D). These are resource-intensive calls which place a significant overhead on the CPU. Therefore, to draw similar objects with slight difference in parameters, GPU Instancing provides much better performance output. In terms of Unity engine, this process can be enabled manually. It is only present on a select few platforms, out of which the below mentioned ones are used:

1. DirectX 11 and DirectX 12 on Windows
2. OpenGL Core 4.1+/ES3.0+ on Windows, macOS, Linux, iOS and Android

Our project environment is setup in Windows 10 operating system and has DirectX 12 installed in it.


Forward Rendering Options

Specular Highlights ✓
Reflections ✓

Advanced Options

Enable GPU Instancii ☐
Double Sided Global ☐

Figure 3.4: GPU Instancing in Unity

3.6 Movement in Virtual Space

A user should be able to move in virtual space according to his will. The movement of the user can be implemented by moving the head in the real world. Whenever the user inclines his head below a certain angle (referred to as the toggle_angle), the movement starts in the direction that the user faces at that moment. This concept has been implemented using the pseudo code mentioned in Algorithm 3.1.

Algorithm 3.1:

```
vector3 current_location;
/*
This function causes the movement of the user in the virtual environment
*/
void vr_walk(float toggle_angle, float speed)
{
boolean move_forward;
if((vr_camera_angle >= toggle_angle)
&& (vr_camera_angle < 90.0))
{
move_forward = true;
}
else
{
move_forward = false;
}
if(move_forward)
{
vector3 forward =
get_current_location();
simple_move(forward,speed);
}
}

/*
This function returns the coordinates of user's current location
*/
vector3 get_current_location()
{
return current_location;
}

/*
This function calculates coordinates of the new location where the user intends to go
*/
void simple_move(vector3 location, float speed)
{
vector3 new_location;
new_location = location * speed;
move_to(new_location);
}

/*
This function changes user's current location by changing the coordinates
*/
void move_to(vector3 location)
{
current_location = location;
}
```

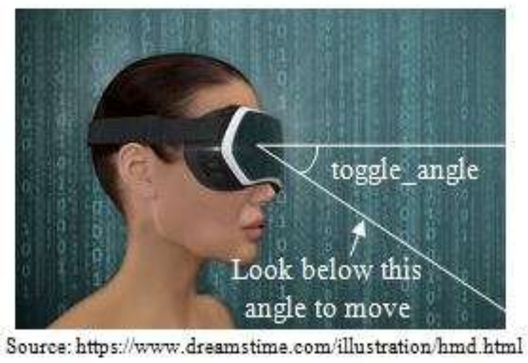

toggle_angle
Look below this angle to move
Source: https://www.dreamstime.com/illustration/hmd.html

Figure 3.5: Movement in Virtual Space

Fig 3.5 shows the toggle_angle as measured from the line of sight of the user. The user will start moving in the virtual world when he/she inclines the head to see below the toggle_angle in the real world.

3.7 Collision-Trigger Model

The collision-trigger model developed by us shows the dynamics of collision between objects and the way in which it can trigger events. An object must be a rigid body so that it follows the laws of physics, so a RigidBody component is added. It is also necessary that the objects contain a collider to detect collisions. Different types of colliders are suitable for different shapes. BoxColliders are available for cubes, SphereColliders for sphere, CapsuleColliders for cylinders while MeshColliders[10] can fit any shape. The physics engine calls functions contained in scripts which are attached to the objects involved in the collision. The code placed in the function is executed in response to the collision event. However, the physics engine can also simply detect the entry of one collider into the space of another without creating a collision. This can be done by configuring a collider as a Trigger (using the Is Triggerproperty).A trigger does not behave like a solid object and will allow other colliders to pass through. The OnTriggerEnter function is called on the trigger object's scripts whenever a collider enters its space. Thus, we can use triggers or collision depending on our goal. If we only need to detect an object entering another object's space then we can use triggers. In case we need actual collision to take place then we use collision. We have used both in our project. Unity provides three methods for collision detection - OnCollisionEnter, OnCollisionStay, OnCollisionExit which are used to define the events that occur when collision begins, collision occurs and when collision ends respectively. Similarly, three other methods like OnTriggerEnter, OnTriggerStay, OnTriggerExit are provided for triggers. We have used Algorithm 3.2 for implementing this model.

```
Algorithm 3.2:
main()
{
Define GameObject gb1, gb2;
/*
add components of rigid body and mesh filter  to gb1, gb2
*/
gb1.AddComponent<RigidBody>();
gb2.AddComponent<RigidBody>();
gb1.AddComponent<MeshFilter>();
gb2.AddComponent< MeshFilter >();
//add collider according to shape of object
addCollider(gb1);
addCollider(gb2);
if(Goal is to only detect object entering a space)
UseTrigger(gb1, gb2);
if(Goal is  actual collision between two objects)
UseCollision(gb1, gb2);
}

/*
Function to add collider according to shape of object
```

```
*/
addCollider(GameObject gb)
{
MeshFilter filter= gb.GetComponent< MeshFilter >();
if(filter.Mesh=”Cube”)
gb.AddComponent< BoxCollider >();
else if(filter.Mesh=”Sphere”)
gb.AddComponent< SphereCollider >();
else if(filter.Mesh=”Cylinder”)
gb.AddComponent<CapsuleCollider>();
else
gb.AddComponent< MeshCollider >();
}

UseTrigger(GameObject gb1,GameObject gb2)
{
Set property IsTrigger=true for gb1’s collider;
//Add script for detecting collision
gb1.AddComponent< Script >();

if (gb2 enters gb1 space)
Use OnTriggerEnter() to define steps
on entering trigger;

while (gb2 is in gb1 space)
Use OnTriggerStay() to define steps
when inside trigger;

if (gb2 exits gb1 space)
Use OnTriggerExit() to define steps
on exiting trigger ;
}

UseCollison(GameObject gb1, GameObject gb2)
{
//Add script for detecting collision
gb1.AddComponent< Script >();

if (gb2 begins collision with gb1)
Use OnCollisionEnter() to define
steps on collision start;

while (gb2 colliding with gb1)
Use OnCollisionStay() to define
steps during collison;

if (gb2 stops colliding with gb1)
Use OnCollisonExit() to define steps
on collision stop;
}
```

3.8 Scripting and Function Calls

Scripting is an essential part of the project. Scripts contain functions which provide the users with results after a successful completion of an activity or interaction with an object. The scripts used in our project include VRWalk as described in Algorithm 3.1, Collision detectors and Event Triggers, scripts for calculating results of physics experiments listed in Table 1, and scripts for interacting with various instruments of the virtual laboratory. Scripts are attached to gameObjects upon which the functions act. Scripts are written in C sharp language. A simple pseudo code for attaching scripts and providing functionality is given in Algorithm 3.3.

Algorithm 3.3

1. Write script with functions to act on gameObjects
2. Attach the script to gameObject
3. Use the sendMessage(“function_name”) function to call a particular function when an event is triggered
4. Handle the event in the function called

The sendMessage()[11] function in Unity is used to call a particular function when an event is triggered. The project is laid out using the Collision-Trigger model which is an extension of the Event-Trigger model. For handling interactivity using LeapMotion, we have included the LeapMotion SDK in Unity. Every gameObject that requires interactivity is attached with a Interaction Manager and an Interaction Behaviour[12] script provided by LeapMotion. The particular gameObject and the script name is mentioned, along with the function to be called.

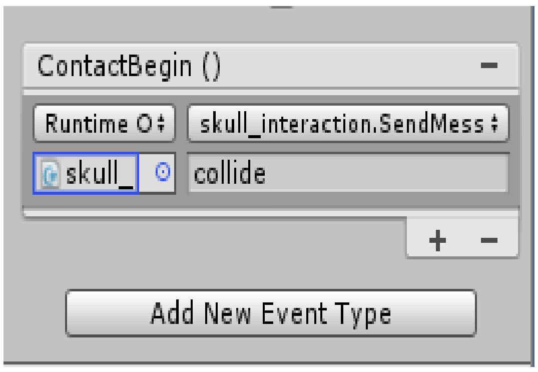

ContactBegin ()
Runtime O | skull_interaction.SendMess
skull_ | collide

Figure 3.6: Script attached to a gameObject

Fig 3.6 contains a script “skull_interaction” attached to a gameObject. The function “collide” is called when the user begins an interaction activity with that gameObject. The ContactBegin() function is provided by LeapMotion’s Interaction Behaviour.

3.9 Deployment Methodology

The block diagram presented in Fig 3.7 depicts the deployment process of the application.

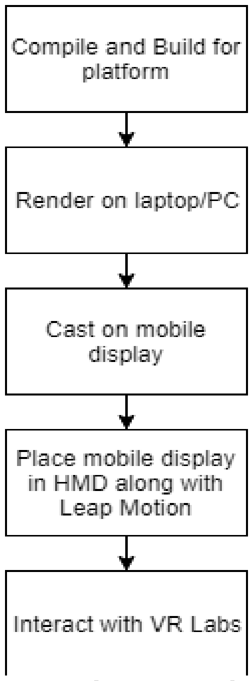

Compile and Build for platform
Render on laptop/PC
Cast on mobile display
Place mobile display in HMD along with Leap Motion
Interact with VR Labs

Figure 3.7: Deployment Methodology

In the initial stages, we are required to compile and build the entire project. If the project has a lot of scripts and assets, the building section will require some time to build. This can be overcome by using the cloud support offered by Unity. After compiling and building the project, we will have to render the VR application on the laptop. This is because the Leap Motion device requires to be connected to the laptop and cannot be interfaced directly with Android. The rendered application will then be cast on the screen of the mobile which will be placed in the Head Mounted Display (HMD). The user will then be able to finally interact with the VR labs and equipment.

IV. CONCLUSION

The main stumbling block in VR becoming mainstream is the high costs associated with the hardware required to run VR applications.Thus, we have created a virtual laboratory with the use of minimum hardware, so that the institutions can provide better opportunities to the students at lower costs. The major drawback of this laboratory is the practice required to handle the hardware. Since the hardware is very inexpensive, the range available for gesture detection is very small and effective usage of the laboratory requires large amount of practice.This application can be further expanded to include a wide range of practicals from different fields of Science and Technology.

ACKNOWLEDGEMENT

We would like to extend our sincere thanks to all people without whom this development would not have possible. We are highly indebted to our prestigious institution VJTI, which has groomed us over the last few years and have provided us with various opportunities including this one. We would like to express our gratitude to our mentor Dr. S. G. Bhirud for sharing his experience with us and guiding us during our difficulties. He also introduced us to the startup Parallax Labs LLP and we are very thankful to him for this helpful gesture. We would also like to mention the part played by 'Parallax Labs LLP' in our project. The founders of the startup guided us during our initial phase and helped us choose the correct technologies from the abundant resources available in the field of Virtual Reality.

REFERENCES

[1]. https://www.vrs.org.uk/virtual-reality/what-is-virtual-reality.html
[2]. https://en.wikipedia.org/wiki/Google_Cardboard
[3]. Anitha.A, Iswariya.K and Karunya.S, A Survey on Next Generation in Revolution - Leap Motion,http://www.ijtrd.com/papers/IJTRD4208.pdf
[4]. https://www.blender.org/about/
[5]. https://en.wikipedia.org/wiki/Stereoscopy
[6]. https://en.wikipedia.org/wiki/Geometry_instancing
[7]. https://docs.unity3d.com/Manual/GPUInstancing.html
[8]. https://docs.unity3d.com/Manual/DrawCallBatching.html
[9]. Peng Hu and Kai Zhu, Strategy research on the performance optimization of 3D mobile game development based on Unity, http://www.jocpr.com/articles/strategy-research-on-the-performance-optimization-of-3d-mobile-game-development-based-on-unity.pdf
[10]. https://docs.unity3d.com/Manual/CollidersOverview.html
[11]. https://docs.unity3d.com/ScriptReference/GameObject.SendMessage.html
[12]. https://github.com/leapmotion/UnityModules/wiki/Getting-Started-%28Interaction-Engine%29
[13].

Anand Deshpande "Learning Physics And Biology In Virtual Laboratory " 'International Journal of Engineering Research and Applications (IJERA) , vol. 8, no.9, 2018, pp 54-60