Code Visualization using VR

Project Mentor: Dr. Sumit Kalra

Anurag Shah
B16CS034
CSE, IIT Jodhpur
anurag.2@iitj.ac.in

Chinmay Garg
B16CS041
CSE, IIT Jodhpur
garg.3@iitj.ac.in

***Abstract*—**

This document serves as the means to guide the readers about the titled BTech Technical Project for the Semester V of Computer Science and Engineering discipline.

***Keywords*—**

Virtual Reality, Debugging, Unity3d, Android, GNU project debugger, Time utility function in Linux

I. Introduction

Debugging is the process of identifying and removing bugs from a program. It has now become an inseparable part of the process of programing. Thus, it is taught compulsorily in every engineering course along with the basic programing skills course. Our project intends to implement this debugging procedure in a virtual space using the virtual reality technologies available nowadays.

Through virtual reality, we will be creating a three-dimensional, computer generated environment which can be explored and interacted with by the user. This virtual environment will be having multiple screens displaying various data regarding the code and its debugging.

This will help users to visualize the procedure of debugging effectively and efficiently, hence, increasing the understanding of the procedure and the code itself. A virtual space also provides a much larger viewable screen, when compared to other conventional computer desktop screens. It has been many a times seen that programmers work with multiple desktops to increase the ease and comfort while debugging. A virtual space eliminates such requirements of multiple screens, as it provides a 360-degree view, in which we can place multiple virtual screens or a single 360-degree curved screen. Hence, this project is useful for both students and programmers.

Our project also aims to provide the users with additional useful and relevant information regarding the code and the debugging procedure that is not generally provided by other debugging softwares.

II. Resources Used

A. *Unity3d: Version 2018.2.7f*

Unity is a cross-platform real-time engine developed by Unity Technologies. Unity gives users the ability to create games in both 2D and 3D, and the engine offers a primary scripting API in C#, for both the Unity editor in the form of plugins, and games themselves, as well as drag and drop functionality.

B. *GDB (GNU Project Debugger)*

GDB is a portable debugger that runs on many Unix-like systems and works for many programming languages. GDB offers extensive facilities for tracing and altering the execution of computer programs. The user can monitor and modify the values of programs' internal variables, and even call functions independently of the program's normal behavior.

C. *Time command in Linux*

This is a useful command for developers who want to test the performance of their program or script. It has a large number of formatting options like: elapsed time, number of page faults, CPU usage, etc.

D. *Google Cardboard*

Google Cardboard is a virtual reality (VR) platform developed by Google for use with a head mount for a smartphone. To use the platform, users run Cardboard-compatible applications on their phone, place the phone into the back of the viewer, and view content through the lenses.

E. *Google Android SDKs*

We will be using the software development kit provided by the Google for the game engine Unity for developing our Cardboard application.

F. *VR Headset*

A virtual reality headset is a head-mounted device that provides virtual reality for the wearer using a smartphone.

III. COMPONENTS OF UNITY3D

Inside the Unity3D, we have majorly used the UI elements provided by the software. Layers of such UI elements were inlaid inside the virtual space to construct our multiple screens as described in the Introduction. The key UI elements used are:

A. *Canvas*

The Canvas is the area that all UI elements should be placed inside, in other words, the Canvas is a Game Object with a Canvas component on it, and all UI elements must be children of such a Canvas.

B. *Panel*

A Panel is a GameObject which does not have its own class, but have three components:

i. Rect Transform
ii. Canvas Renderer
iii. Image Script

In our project, we have used this component to place all our texts.

C. *ScrollRect*

A ScrollRect can be used when content that takes up a lot of space needs to be displayed in a small area. The ScrollRect provides functionality to scroll over this content. We have combined ScrollRect with a Mask in order to create a scroll view, where only scrollable content inside the scroll view is visible. We have also combined two ScrollViews to show the constrained text using horizontal and vertical scrollbars.

D. *UI Text*

The Text component, used to render text onscreen for various purposes such as labels, buttons, and other information is called UI Text. We have used this to show code, output etc. in execution window.

E. *FPS Controller*

This is a First Person Controller, which gives the user an experience as if he is present inside the game and allows 360 degree view of the scene. Gyroscope is used to track the movement of the head, so that the user gets a near-reality experience.

F. *Slider*

A standard slider that can be moved between a minimum and maximum value. When a change to the slider value occurs, a callback is sent to any registered listeners of Slider.onValueChanged. We have used it to show the amount of CPU usage, page faults and Memory usage.

G. *C# Code*

Apart from all these we also will be using C# script to change the code as and when the corresponding function is called.
At present we have hard coded code in the C# script. The code to the same is as below:

```
using System.Collections;
using System.Collections.Generic;
using System.Text;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;

public class ChangePointer : EventTrigger {
    Text code;
    static int codeno = 0
    void Start () {
        code = GetComponent<Text> ();
        main = code.text;
    }
    void Update () {
        if (Input.GetKey (KeyCode.Space)) {        }
    }

    public override void OnPointerClick(PointerEventData data) {
        if (codeno == 0) {
            Debug.Log ("OnSelect called");
            Text code = GetComponent<Text> ();
            code.text = "void test_function() {
                \n cout << \"Inside function\"; \n} ";
            codeno = 1;
        } else {
            code.text = "Second function";
            codeno = 0;
        }
    }
}
```

IV. EXTRACTING THE INFORMATION

For our project we need the following information texts, to display on the screens present in our virtual space:

A. Input Program Code

We need to display the code which we are debugging. It is the code for which we wish to visualize the debugging process through our project.

B. GDB Input Commands

We need to know the input commands that were given to the GDB during the debugging of the required program code. This can be done using file handling. All the input commands can be prestored in a text file called inputgdb.txt and can be given as an input to the GDB.

```
g++ -g main.cpp
gdb a.out < inputgdb.txt
```

C. GDB Output

The output can be saved in a logging file using the command:

```
set logging on
```

This has to be written before writing the debug commands. It will store the result of all the following GDB commands in a file named gdb.txt.

D. Time Statistics

The time command is a utility feature of Linux, which runs the specified program *command* with the given arguments. When *command* finishes, time writes a message to standard error giving timing statistics about this program run.

The statistics consist of the elapsed real time, the user CPU time, the system CPU time, and other time, memory and i/o information when specified the format tag.

This is the code used to retrieve time statistics:

```
/usr/bin/time -o time.txt -f "\nCPU
seconds used by user = %U \nCPU seconds
used by kernel = %S \nInputs = %I
\nOutputs = %O\nPage Faults = %F \nTotal
Memory used (KB) = %K \nCPU percentage
used = %P \n"
```

Output:

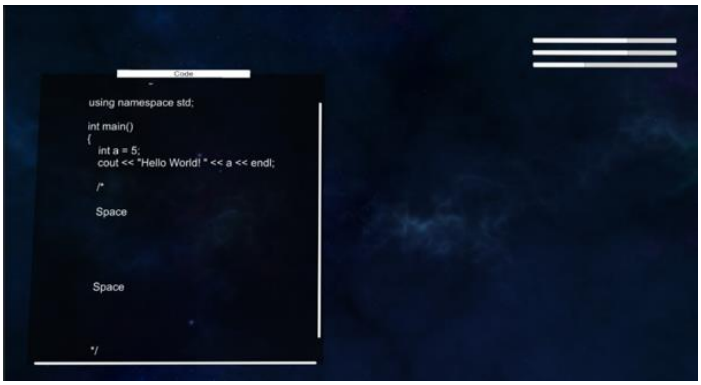
```
CPU seconds used by user = 0.98
CPU seconds used by kernel = 0.56
Inputs = 0
Outputs = 0
Page Faults = 0
Total Memory used (KB) = 0
CPU percentage used = 91%
```


Acknowledgment

We wish to sincerely thank our mentor, Dr. Sumit Kalra sir for his unconditional and constant help and support. Whether it be helping us to understand the project, procurement of the VR headset, or guiding us through the process, he was always there.

Images of the Virtual Space

A. Slider and UI Text elements



B. Multiple Screens

REFERENCES

The following are not the citations, but the references we used throughout the project, as for this documentation we did not need to refer any publication:

i. https://docs.unity3d.com/Manual/UISystem.html
ii. https://stackoverflow.com/
iii. https://unity3d.com/learn/tutorials
iv. https://youtu.be/MWOvwegLDl0
v. https://www.youtube.com/watch?v=HQ8Tttcksu4