TIMING FUNCTIONS CITY

Edgar Velazquez

CS404 Program Monitoring and Visualization

Technical Report

1. Introduction

For my semester project, I committed to create a software city that would be able to visualize the times spent on each procedure. This paper will give a brief explanation on how to use my tool, a description of what it does, and what I've been able to observe while using it.

2. Guide

The first step to getting started will be to extract the *.zip* file. Once this is done the user will notice a *make.bat* file. Executing this file will put everything together and create a **hw6** executable. The user will then need to import a program to monitor and its corresponding *.dat* file. The imported program will then need to be compiled and passed to hw6 as follows: *"./hw6 programName <datFile -t=x -pos=XxYxZ"*. Two optional arguments are being passed in this example. The first is *"-t"* for time. This argument will let the user select the time in milliseconds for each block placed on the building. The next is "-pos". This will allow the user the enter a (XYZ) position to spawn upon initial execution. Notice, if no arguments are specified defaults are 100ms for time, and 0x0x0 for the initial position. Upon spawning, the user is able to move around the city and control the speed of the monitoring. The controls are as follows:

Key	Action
W	Move Forward
S	Move Back
Arrow Keys	Left, Right, Up, & Down Camera Movements
P	Increase Monitoring Speed
O	Decrease Monitoring Speed
Q	Quit

After a user quits the program, a log will be written to the terminal showing the exact time spent on each procedure. If a user forgets the controls, pressing any key that is not shown on this table will trigger a log to the terminal showing the user the controls.

3. Description

My tool can draw a small section of a city that is composed of different procedures shown as buildings. When a user runs the program, it is placed at an intersection of a road with buildings to the left and right. The user will then notice that these building have names on the first level of the building and grow dynamically. The user will also notice that some of them light up at different brightness levels.

The explanation for all this is the following. As the user spawns, they will want to focus only the buildings that are to the left and right of them. These building are purposely placed there. In the left are all the functions which are built in, and to the right are all the procedures which are user-defined. This information is statically obtained and rendered to the screen. Once the monitoring starts the program keeps track of the calls being made to each procedure and records their time. By default, every cube represents 100ms spent on a procedure, so If a

procedure spends 330ms it will be composed of 4 levels. The first is by default rendered with the name of the procedure, and 1 level will be added for every 100ms spent in the function. The bottom level will also light up white each time that function is called. The white might appear very dim at first, this is because the more calls to that function the brighter the white gets. Lastly when a user presses “q” to stop the program, he will notice there is a log on the console to show the exact time spent on all the functions and procedures. This was added because the cubes are not able to represent the exact time spent on each function, they will only show blocks of time.

4. **Observations**

   With my time using this tool, I was able to learn new information. The first thing I noticed was that significantly more time was spent in functions that are built in compared to user-defined ones. Built in functions also spend more time when they are called because I noticed the cubes were still very dim in brightness and the buildings were 10+ stories higher than the user-defined procedures when using the default arguments. The next interesting thing was being able to see how many functions were not being called at all during the program execution.

5. **Conclusion**

   The goal of my city is to help the user see how the program moves from function to function and show the time is spent on each function as its moving. This can help the user see if there are one or more functions that can be modified in order to help the execution of the program.