Simple Tool for Analyzing Non-exotic Datatypes (STAND)

Nick Avery, James Young

CS 504 – ST: Program Monitoring and Visualization

May 10, 2019

Introduction

While memory may no longer be the concern of many programmers with todays computers, it is a concept that certainly still occurs and may be of interest to some. The tool described in the following paper is an approach to visualizing the memory taken up by common data types found in the Unicon language, with the intent to show its behavior over the course of a program's runtime.

Purpose

Task

The goal of this tool is to help show various aspects of memory use within a Unicon program. Ideally, this tool is used to get a better understanding of what the program consists of in memory, when certain common data types/structures are being used, and how these common types/structures relate to each other in both size and time.

Audience

Those programming in Unicon and who are interested in finer details of a programs usage of memory would find the most use in this tool. Additionally, while not the intended audience, there could be a potential for an educational use, helping to show a more tangible representation of memory.

Target

STAND focuses primarily on memory allocation events, most specifically a group of common data types: lists, tables, strings, reals, records, and sets. From these events, both the frequency and the size of the allocations are recorded, and then used to create the visualization.

Representation

6 cones grow from a central sphere, with each cone representing one of the common data types. From this cone, one can easily see the memory size of types, both in and of itself, and in relation to other types. Additionally, if a cone has a flat or cylindrical section, one can infer that no allocations of that type occurred in that time period.

Medium

Unicon's 3D facilities are used to render this visualization.

Key Features

Event Slice Control

As an attempt to handle varying length programs, in that some run for a short amount of time, and others much longer, a control was implemented to determine when a section of the cylinders is to be drawn. The default configuration for this is 200 events, meaning that every 200 allocation events, the next section of all 6 cylinder will be drawn, even if nothing was allocated for that type. What this means is that you can technically go down to single allocation events and get extremely fine-grained information, at the cost of the visualization running slower/longer.

Log Base Control

Similar to the control of the event slice, the visualization utilizes a log scale to help control the size of the visualization. The default setting of the log base is base 10, but the user can specify if they would like to use a different base, should they believe it would help them find what they are looking for. Additionally, should the user want it, the option to not use log scaling is present. If the value -1 is used for the log base, the program will instead use raw bytes to handle the size of the cylinder's radius (this can get ugly quickly).

Positioning of Memory Cones

The memory cones in the visualization are not static in their position. While each common type has a resting area, indicated in the accompanying Info Screen, every cone can be selected and moved elsewhere. This is done by selecting a cone via click, and then using either 1-6 to potentially move it to another cone's position, or by using keys U-J, I-K, and O-L for finer control. The main advantage of this scheme is the ability to control the visualization to fit the users needs. One interesting feature that came from testing is positioning two cones at the same location. Because the two shapes overlap each other, it becomes very clear when the values are either close to each other, or one overtakes the other.

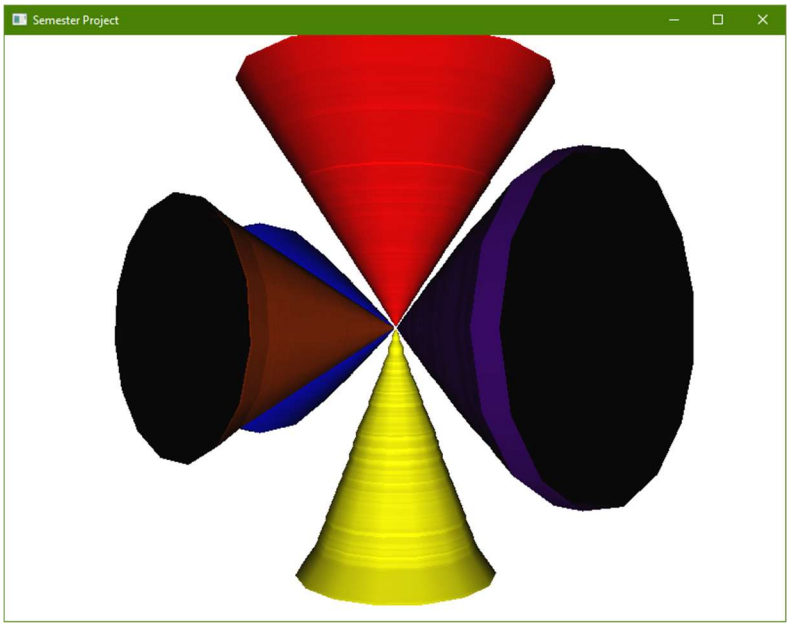
Additional Window Display

In addition to the main visualization window, there is a secondary window that contains more text-based information. First visible would be the color legend, as well as the hotkeys for the default positions of the cones. However, upon clicking any of the cones, additional information will show up. Both the total memory of the data type, as well as the event number representing when it was most recently allocated will be displayed. This is to hopefully provide multiple granularities to the same information, and ultimately provide more useful information to the user.

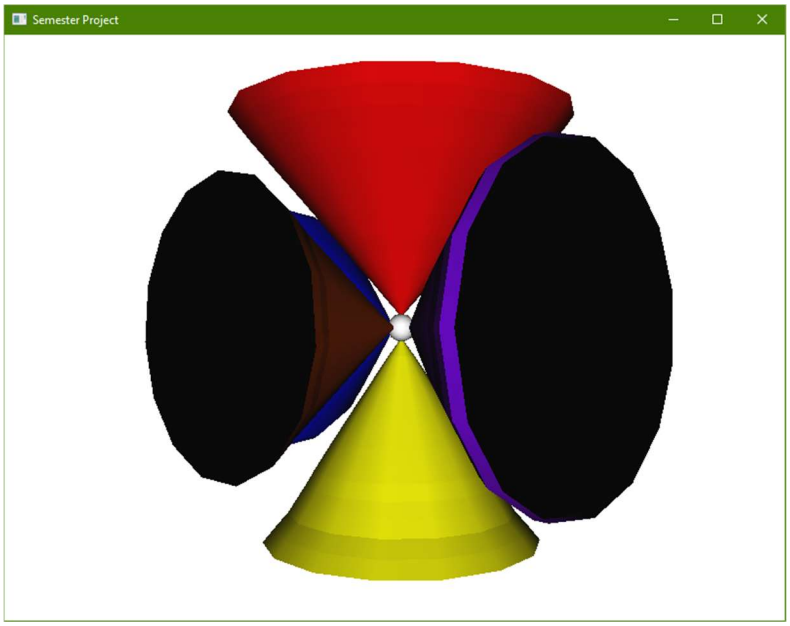
Examples

Event Slice Control

In the following two images, we have the tool set first to a 200-event time slice, and then to a 2000-event time slice. Note the difference in detail.



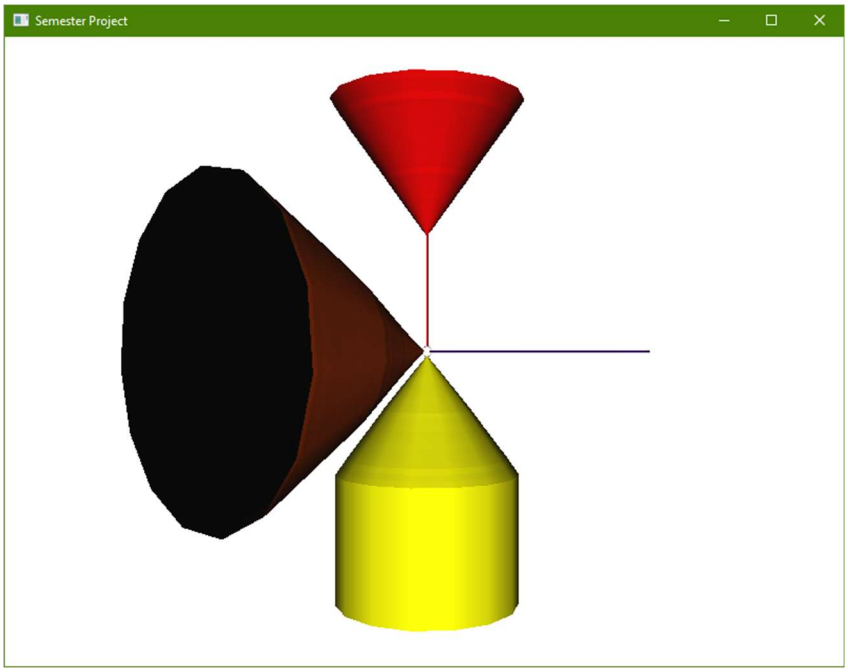
STAND run on the beards program with default parameters



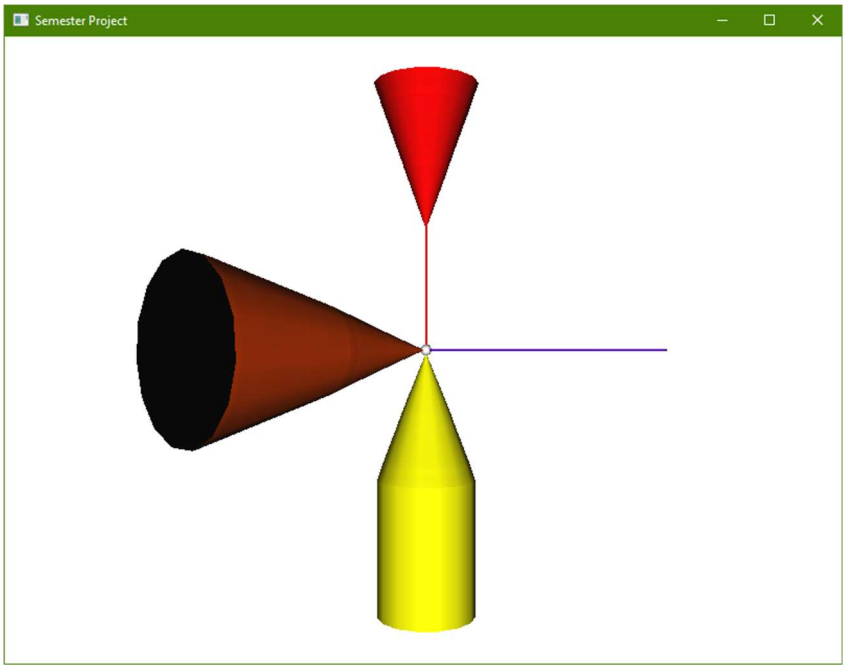
STAND run on the beards program with argument `-ts=2000`

Log Base Control

To show log base functionality, the base parameter of log base 10 and the provided log base 100 were used. Note the difference in cone size.



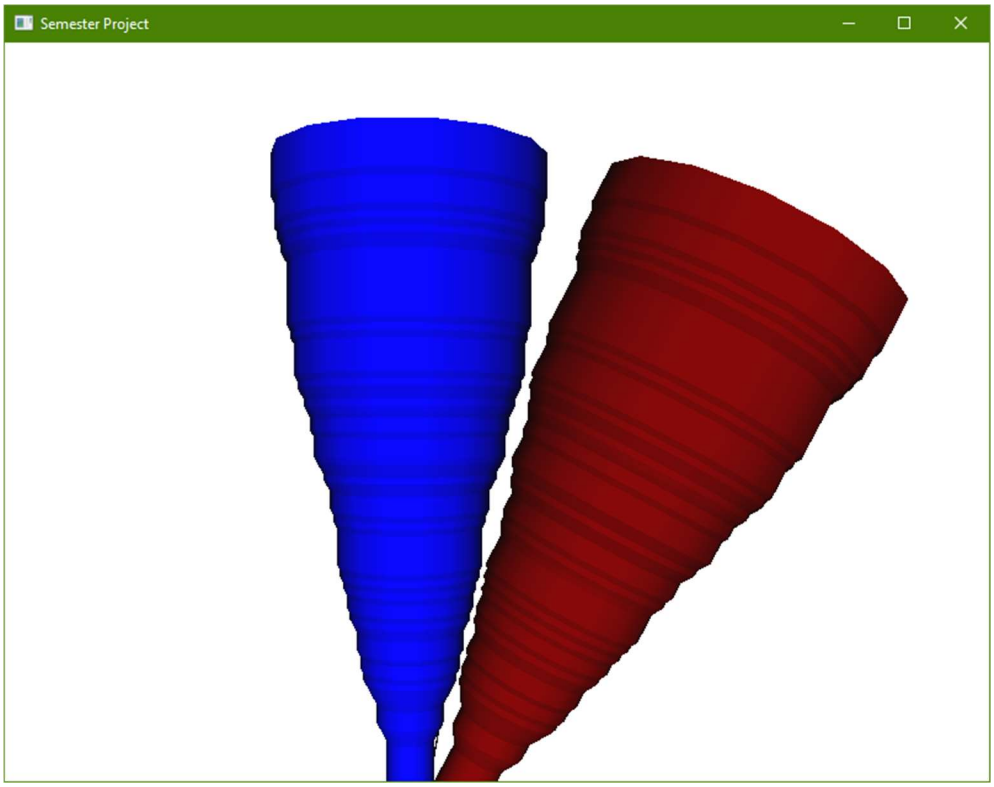
STAND run on the contbl program with default parameters



STAND run on the contbl program with argument `-lb=100`

Positioning of Memory Cones

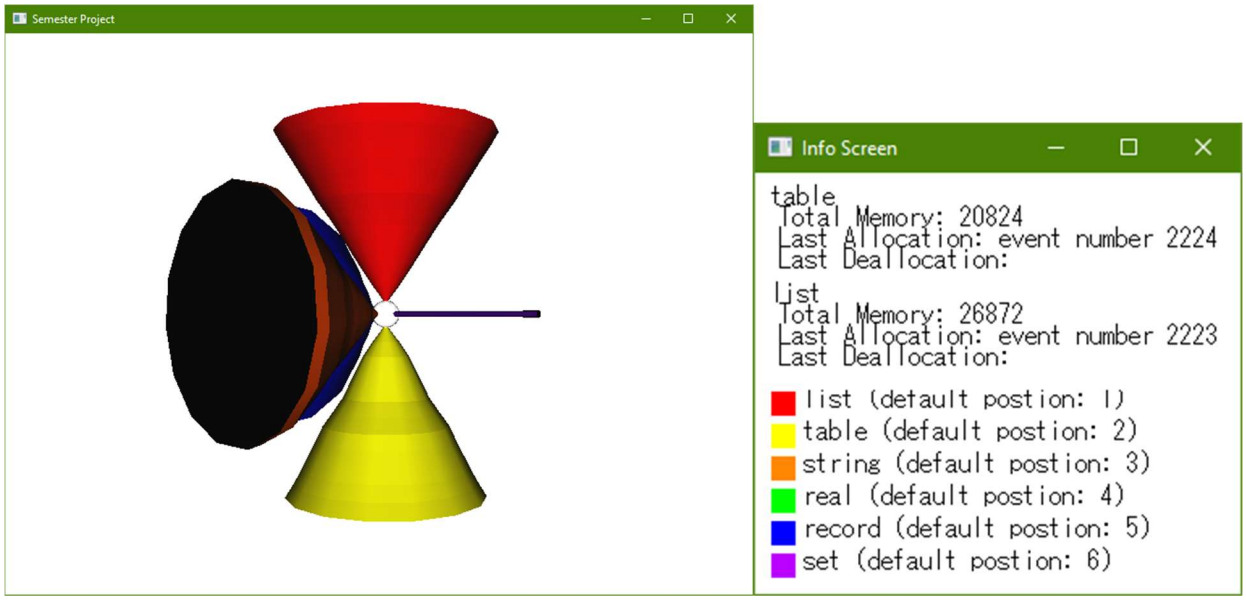
Another useful feature is the ability to move the memory cones around. In this example, the list cone (red) has been moved close to the record cone (blue) and it becomes there is an apparent relation between the shapes of the two objects, implying that they are likely allocated in a very similar manner and timeframe.



Using movement hotkeys to better align cones for comparison from the poem program

Additional Window Display

Sometimes, two data types might be very similar in shape and size, but a desire exists to know, down to the byte, which is larger. With the use of the Info Screen, more detailed information is provided.



Comparing tables and lists found in the chess program

Evaluation

In evaluating this program, a small set of test programs were used: beards, chess, contbl, and poem. Some notable features that have cropped up while testing and putting together this document primarily pertain to how the memory cones are shaped. I was pleasantly surprised to see how clearly it depicts / how easy it is to see when data types are no longer being allocated and when they just begin to start getting allocated. Additionally, having similarly shaped cones was a neat feature that was technically unintended, but came out by design of the project. This combined with the features that the program already exhibits (such as the info screen) makes me feel that this program has at least achieved something in providing information through visualization.

Evaluating on other parameters, such as scalability and performance, the tool performs reasonably well. While developing the tool, it became rather apparent that measure would have to be taken to speed up the runtime of the visualization. Optimizations from handling when and where objects were being drawn, as well as how many refreshes were being performed certainly helped, but the most helpful towards this task was implementing the event slice control. The benefits for having event slices is twofold, in that it reduces the amount of shapes needed to be drawn overall, as well as compacting down larger programs into a smaller amount of shapes.

The occlusion problem still exists within this tool, in that it is almost impossible to see all 6 cones all at once in a meaningful way in the default configuration. Our solution to this was to simply be able to control the position of the cones around the sphere, and to a certain degree, it has worked. The hotkeys do indeed provide a means of manipulating the cones' position, with the only downside is that they feel a little clunky.