Setup on participant's local machine

Introduce participants to STLViz's features

Task - Bug Finding - 30-60 mins hopefully
You will be given a small buggy C++ 17 program designed to solve an input, output problem similar to interview questions.
You have access to the source code, a build command, and an execute command.
Your goal is to identify the bug and the location (line number and general offset) it occurs.
The input and expected output for a single testcase will be provided, and the program will take that testcase as input and output an incorrect answer.
You are allowed to edit and recompile the program, but you do not need to fix the program.
Rebuilding the program should take only a few seconds.
There will be 6 total questions, you will solve three without the visual debugger and three with the visual debugger in alternating order (start with random).
After each problem, you will rate the difficulty of the problem (1-10) (you can change answers of prior ratings).
You will also be timed, but don't feel pressured to solve as fast as possible.
timer starts after program executed or source code opened, can spend as long on problem as needed before

Start with sample question
P0 - VEC sum of numbers - starts at index 1

Tasks
Random order, 3 with STLViz 3 without in alternating order starting without
P1 - VEC bsearch kth largest - round up instead of down
P2 - VEC psum query - off by 1 error
P3 - BTREE tree longest path - off by 1 for ^ shape
P4 - VEC kadanes max absolute subarray sum - max instead of min
P5 - SET find overlap of two arrays - use size of b instead of size of a
P6 - STACK next higher element to left - stack uses <= instead of <

Source Code Collection
Participant sends a zip of the user_study directory for later analysis

Background Questionnaire
Years of experience with C++
Years of experience with programming
Primary programming languages / frameworks (choose at most 3)
Main field of study/work
Academic year (Freshman, Sophomore, Junior, Senior, Graduate, MS, PhD, other)
Preferred method of debugging smaller programs (debugger, IDE breakpoints, print statements, etc.)

Likert Questionairre
1=Strongly Disagree
2=Disagree
3=Somewhat Disagree
4=Neither Agree nor Disagree
5=Somewhat Agree
6=Agree
7=Strongly Agree

Post Study
Question 1 - STLViz's UI is intuitive to use
Question 2 - STLViz's UI is overwhelming
Question 3 - STLViz is effective for debugging small programs
Question 4 - I have practiced solving these types of problems
Question 5 - STLViz reduced the mental load required to debug
Question 6 - In this session, I found debugging with STLViz easier than debugging without
Question 7 - STLViz has potential for more features than text-based debuggers
Qeustion 8 - I regularly use text-based debuggers (like gdb)
Question 9 - I regularly use visual debuggers
Question 10 - I can see myself using visual debuggers regularly in the future

Semi-structured Interview
What features of STLViz did you find most effective
What features of STLViz did you find least effective
What are some features you would want that STLViz currently lacks
What usecases do you think STLViz is best designed for
What difficulties did you have during the study that could affect the results
Any other thoughts on STLVIZ?

Sujay Sudarsan

Background Questionnaire
Years of experience with C++
~3

Years of experience with programming
~6

Primary programming languages / frameworks (choose at most 3)
C++ Java Python

Main field of study/work
SWE

Academic year (Freshman, Sophomore, Junior, Senior, Graduate, MS, PhD, other)
Graduate

Preferred method of debugging smaller programs (debugger, IDE breakpoints, print statements, etc.)
print statements

Task Order
num_smaller - stlviz - 14:43
7
max_subarr - nostlviz - 13:54
6
max_path - stlviz - 12:06
4
nearest_higher - nostlviz - 7:12
3
sum_queries - sltviz - 4:00
1
intersect_arrays - nostlviz - 1:47
1
Likert Questionairre
1=Strongly Disagree
2=Disagree
3=Somewhat Disagree
4=Neither Agree nor Disagree
5=Somewhat Agree
6=Agree
7=Strongly Agree

<u>Post Study</u>
Question 1 - STLViz's UI is intuitive to use - 6
Question 2 - STLViz's UI is overwhelming - 1
Question 3 - STLViz is effective for debugging small programs - 7
Question 4 - I have practiced solving these types of problems -5
Question 5 - STLViz reduced the mental load required to debug - 6
Question 6 - In this session, I found debugging with STLViz easier than debugging without - 5
Question 7 - STLViz has potential for more features than text-based debuggers - 4
Qeustion 8 - I regularly use text-based debuggers (like gdb) - 3
Question 9 - I regularly use visual debuggers - 1
Question 10 - I can see myself using visual debuggers regularly in the future - 5

Semi-structured Interview
1) What features of STLViz did you find most effective

reverse execution

2) What features of STLViz did you find least effective

tab filters

3) What are some features you would want that STLViz currently lacks

custom filters / search
snap to area with visualized stuff to prevent getting lost

4) What use cases do you think STLViz is best designed for

studying algorithmic problems

5) What difficulties did you have during the study that could affect the results

nothing

6) Any other thoughts on STLVIZ?

Would have been easier to debug with more experience with the visualizer(Q3)

—----------

Aditya

Background Questionnaire
Years of experience with C++
5

Years of experience with programming
6

Primary programming languages / frameworks (choose at most 3)
C++, Python, SQL

Main field of study/work
Data Scientist

Academic year (Freshman, Sophomore, Junior, Senior, Graduate, MS, PhD, other)
Graduate

Preferred method of debugging smaller programs (debugger, IDE breakpoints, print statements, etc.)
Print statements, dry run

Task Order
num_smaller - stlviz - 16:06
6

max_subarr - nostlviz - 7:54
2

max_path - stlviz - 6:39
2

nearest_higher - nostlviz - 5:35
3

sum_queries - sltviz - 2:58
1

intersect_arrays - nostlviz - 3:00
1

Likert Questionairre
1=Strongly Disagree
2=Disagree
3=Somewhat Disagree
4=Neither Agree nor Disagree
5=Somewhat Agree
6=Agree
7=Strongly Agree

Post Study
Question 1 - STLViz's UI is intuitive to use - 5
Question 2 - STLViz's UI is overwhelming - 2
Question 3 - STLViz is effective for debugging small programs - 5
Question 4 - I have practiced solving these types of problems - 6
Question 5 - STLViz reduced the mental load required to debug - 4
Question 6 - In this session, I found debugging with STLViz easier than debugging without - 2
Question 7 - STLViz has potential for more features than text-based debuggers - 4

Qeustion 8 - I regularly use text-based debuggers (like gdb) - 1
Question 9 - I regularly use visual debuggers - 1
Question 10 - I can see myself using visual debuggers regularly in the future - 5

Semi-structured Interview
1) What features of STLViz did you find most effective
Clear and Intuitive visuals, easier to track variable updates

2) What features of STLViz did you find least effective
Tabs

3) What are some features you would want that STLViz currently lacks
Want to directly go to output

4) What use cases do you think STLViz is best designed for
Learning purposes, beginner

5) What difficulties did you have during the study that could affect the results
Visualizer did not work sometimes, had to restart

6) Any other thoughts on STLVIZ?
-

—----------

Nathaniel

Background Questionnaire
Years of experience with C++
0

Years of experience with programming
3

Primary programming languages / frameworks (choose at most 3)
Java, Python

Main field of study/work

Computer Science Major

Academic year (Freshman, Sophomore, Junior, Senior, Graduate, MS, PhD, other)
Sophomore

Preferred method of debugging smaller programs (debugger, IDE breakpoints, print statements, etc.)
breakpoints

Task Order

Nearest_higher - stlviz - 6:47
3

Num_smaller - nostlviz - 25 : 42
8

Intersect_arrays - stlviz - 4:20
1

Sum_queries - nostlviz - 12:32
4

Max_subarr - stlviz - 5:48
3

Max_path - nostlviz - 9:23
7

Likert Questionairre
1=Strongly Disagree
2=Disagree
3=Somewhat Disagree
4=Neither Agree nor Disagree
5=Somewhat Agree
6=Agree
7=Strongly Agree

Post Study
Question 1 - STLViz's UI is intuitive to use - 6

Question 2 - STLViz's UI is overwhelming - 2
Question 3 - STLViz is effective for debugging small programs - 7
Question 4 - I have practiced solving these types of problems - 2
Question 5 - STLViz reduced the mental load required to debug - 6
Question 6 - In this session, I found debugging with STLViz easier than debugging without - 7
Question 7 - STLViz has potential for more features than text-based debuggers - 6
Question 8 - I regularly use text-based debuggers (like gdb) - 6
Question 9 - I regularly use visual debuggers - 1
Question 10 - I can see myself using visual debuggers regularly in the future - 4

Semi-structured Interview
1) What features of STLViz did you find most effective
The visual representation of arrays/vectors together, easy to tell apart, organized well

2) What features of STLViz did you find least effective
The right side of the visualizer(with the line numbers)

3) What are some features you would want that STLViz currently lacks
An option to watch variables/expressions (like in gdb)

4) What use cases do you think STLViz is best designed for
Best for more complex data structures like binary trees, or different ways to set up arrays.

5) What difficulties did you have during the study that could affect the results
Lack of c++ syntax knowledge

6) Any other thoughts on STLVIZ?
It's cool, thumbs up