Project Overview

Tejal Kulkarni - CS21BTECH11058
Nitya Bhamidipaty - CS21BTECH11041
Deepshikha - CS21BTECH11016
Muskan Jaiswal - CS21BTECH11037

24th November 2023

1 Why we chose this language?

1. The motivation behind designing this language is to combine computation and geometry.
2. Though there are other software for creating diagrams, they do not natively support programming.
3. The idea is to make abstract concepts more intuitive by easing the process of making geometrical figures.
4. We feel that this language will help automate the process of visualizing geometrical figures. An user can integrate external data with code to create diagrams.
5. Our language is mainly intended for high school geometry. We feel that integrating our language with school curriculum can help nurture programming mindset at a young age.

2 Difficulties Faced:

1. Our language allows global statements other than declarations. Hence conversion to C++ was difficult since we had to take care of these statements and put them in int main
2. Our language is mainly for visualisation and hence we made use of OpenGL Library for conversion. Since OpenGL has infinite running loop for displaying diagrams, global variable changes makes the display unpredictable. For this we had to store all display items in a data structure and display later after all changes in state were done.

3. During semantic analysis, while doing dimension checking we needed values of the dimension. Hence we allowed only constant expressions in array declaration.

4. Line array includes "-" symbol which is the same as "-" arithmetic operator , so this caused ambiguity .

3 Pipeline of compilation Process:

1. Lexical Analysis : We have used the flex tool to write regular expressions and keywords. Tokens are generated and passed to the parser.

2. Syntax Analysis : We have used bison tool to write our grammar rules. Syntax errors are seen as errors in stderr.

3. Semantic Analysis : Various static semantic checks are done in this phase. For this symbol_table.hpp and utils.cpp are made to write helper functions needed for semanti

4. Code Generation : Angle is translated into C++ .The translated code is integrated with OpenGL c++ with the help of functions in "standard_lib.hpp"

4 How to run on test cases?

1. Navigate in the "codes" directory.

2. If you want to individually test the cases run the command "make parser" and then run the executable ./parser ¡input file path¿

```
cd codes
make parser
./parser "input file path"
./translate_run.sh
```

3. "seq_tokens_.txt" will be generated for the list of tokens.

4. "translated.cpp" will be generated for the translated C++ code.

5. Output will be seen in a window.