OpenSCAD's Customizer

Submitted for partial fulfilment of the Degree
of
Bachelor of Technology
(Computer Science and Engineering)

Submitted by:
Amarjeet Singh Kapoor
1311017

Submitted to:
Sukhjit Singh Sehra
Training Coordinator
CSE Department

Department of Computer Science & Engineering
Guru Nanak Dev Engineering College
Ludhiana 141006

Abstract

User Interface for Customizing Models in OpenSCAD is the project that I worked upon for my 6-month training and also as Google summer of code project. It is under the umbrella organization of BRL-CAD. OpenSCAD is an open source organization that serves a free software to create solid 3d CAD objects. OpenSCAD has in a way redefined how easy 3D modeling can be. But the Wikipedia article on OpenSCAD says that it is a non-interactive modeler, but rather a 3D compiler based on a textual description language. Pay attention to the above line, its primarily what Ill be talking about.

What the guys over at Wikipedia said is true but their version of the truth needs a little filtration (rather trimming). OpenSCADs way of customization is interactive, just not through a graphical interface. And this contingency makes the whole 3D modeling thing a little less easy than it can be. But all of that is about to change.

Solid 3D modeling. That sounds like some serious business. But its just an awesome tool for making models pertaining to many uses (mostly 3D printing). And 3D printing as we can all agree upon is cool. 3D models can be created by anyone using OpenSCAD. OpenSCAD is as much for designers as it is for you and me. What else can most people agree upon apart from the fact that solid 3D modeling is cool? A graphical interface is simpler and more intuitive to use. There is a general aversion for typing commands in order to get things done. Simply put, more people have an inclination towards GUI.

This is something that OpenSCAD lacked. But the benevolent folks at Thingiverse.com found a way to help out the demographic intersection of GUI lovers and OpenSCAD users. The website provides an easy to use interface to customize models of OpenSCAD. All one needs to do is upload the OpenSCAD file. After uploading the file, what youll see can only be described as being magic. Im kidding, its just very useful is all. The OpenSCADs script is used to make a form containing slide bars, text boxes, combo boxes, labels, etc all for the singular purpose of customizing models.

My project was to include similar functionality into OpenSCAD itself. Constantly having to upload files created in one software (OpenSCAD) to a website in order to customize your models can get a little problematic as one is uploading scripts without being able to confirm how the script will translate into a form on the site. Wouldnt it be great if everything is at one place, the original place: OpenSCAD? Of course, it would.

My project intends to define a user interface to customize models interactively instead of having to modify them manually. It will enable the user to create the templates for a given model which can further be changed as per users requirements.

This project will allow the modelers to create generic models (templates) which others can then customize to cater to their own use.

Acknowledgement

I, student of Guru Nanak Dev Engineering College, Ludhiana, have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

The author is highly grateful to Dr. M.S. Saini Director, Guru Nanak Dev Engineering College, Ludhiana for providing him with the opportunity to carry out his Six Weeks Training at Testing and Consultancy Cell, Guru Nanak Dev Engineering College, Ludhiana.

The author would like to wholeheartedly thank Dr. H.S. Rai Dean, Testing and Consultancy Cell, Guru Nanak Dev Engineering College, Ludhiana who is a vast sea of knowledge and without whose constant and never ending support and motivation, it would never have been possible to complete the project and other assignments so efficiently and effectively.

Finally, I would thank's My Mentors at OpenSCAD organization Marius Kintel and Torsten Paul. Without their encouragement and Guidence, it would not have been possible to complete this project in such an efficient manner.

Amarjeet Singh Kapoor

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION OF ORGANIZATION

Figure 1.1: Guru Nanak Dev Engineering College

I had my Six Month Industrial Training at TCC-Testing And Consultancy Cell, GNDEC Ludhiana. Guru Nanak Dev Engineering College was established by the Nankana Sahib Education Trust Ludhiana. The Nankana Sahib Education Trust i.e NSET was founded in memory of the most sacred temple of Sri Nankana Sahib, birth place of Sri Guru Nanak Dev Ji. With the mission of Removal of Economic Backwardness through Technology Shiromani Gurudwara Parbandhak Committee i.e SGPC started a Poly technical was started in 1953 and Guru Nanak Dev Engineering College was established in 1956.

The main goal of this institute is:

- To build and promote teams of experts in the upcoming specialisations.
- To promote quality research and undertake research projects keeping in view their relevance to needs and requirements of technology in local industry.
- To achieve total financial independence.
- To start online transfer of knowledge in appropriate technology by means of establishing multipurpose resource centres.

1.1 Testing and Consutancy Cell

I had my Six Month Institutional Training at TCC i.e Testing And Consultancy Cell, GNDEC Ludhiana under the guidance of Dr. H.S.Rai Dean Testing and Consultancy Cell. Testing and Consultancy Cell was established in the year 1979 with a basic aim to produce quality service for technical problems at reasonable and affordable rates as a service to society in general and Engineering fraternity in particular.

Consultancy Services are being rendered by various Departments of the College to the industry, Sate Government Departments and Entrepreneurs and are extended in the form of expert advice in design, testing of materials & equipment, technical surveys, technical audit, calibration of instruments, preparation of technical feasibility reports etc. This consultancy cell of the college has given a new dimension to the development programmers of the College. Consultancy projects of over Rs. one crore are completed by the Consultancy cell during financial year 2009-10.

Ours is a pioneer institute providing Consultancy Services in the States of Punjab, Haryana, Himachal, J&K and Rajasthan. Various Major Clients of the Consultancy Cell are as under:

- Northern Railway, Govt. of India
- Indian Oil Corporation Ltd.
- Larson & Turbo.
- Multi National Companies like AFCON & PAULINGS.
- Punjab Water Supply & Sewage Board

CHAPTER 2
INTRODUCTION TO PROJECT

2.1 Overview

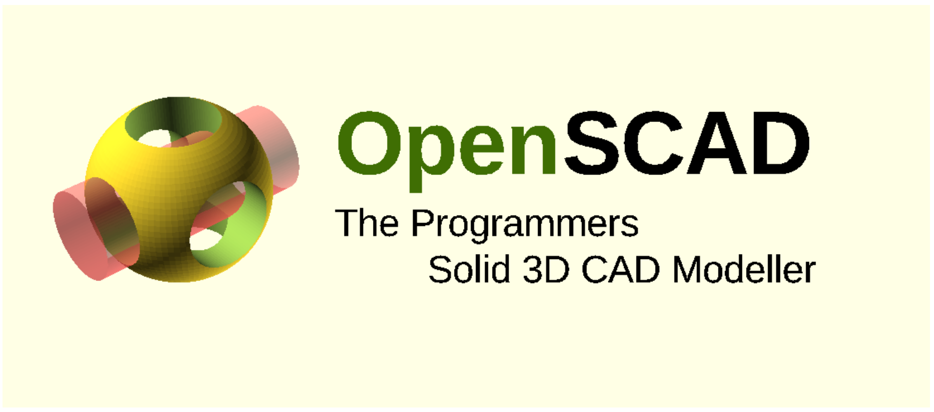


Figure 2.1: OpenSCAD's logo

User Interface for Customizing Models in OpenSCAD is the project that I worked upon for my 6-month training. It is under the umbrella organization of BRL-CAD. OpenSCAD is a free and Open-source software application for creating solid 3D CAD objects. It is a script only based modeler, with a specific description language. Parts cannot be selected or modified by mouse in the 3D view. An OpenSCAD script specifies geometric primitives and defines how they are modified and manipulated to render a 3D model. OpenSCAD is available for Windows, Linux and OS X. It does constructive solid geometry (CSG).

OpenSCAD has in a way redefined how easy 3D modeling can be. But the Wikipedia article on OpenSCAD says that it is a non-interactive modeler, but rather a 3D compiler based on a textual description language. Pay attention to the above line, its primarily what Ill be talking about.

Solid 3D modeling. That sounds like some serious business. But its just an awesome tool for making models pertaining to many uses (mostly 3D printing). And 3D printing as we can all agree upon is cool. 3D models can be created by anyone using OpenSCAD. OpenSCAD is as much for designers as it is for you and me. What else can most people agree upon apart from the fact that solid 3D modeling is cool? A graphical interface is simpler and more intuitive to use. There is a general aversion for typing commands in order to get things done. Simply put, more people have an inclination towards GUI.

This is something that OpenSCAD lacked. But the benevolent folks at Thingiverse.com found a way to help out the demographic intersection of GUI lovers and OpenSCAD users. The website provides an easy to use interface to customize models of OpenSCAD. All one needs to do is upload the OpenSCAD file. After uploading the file, what youll see can only be described as being magic. Im kidding, its just very useful is all. The OpenSCADs script is used to make a form containing slide bars, text boxes, combo boxes, labels, etc all for the singular purpose of customizing models.

My project was to include similar functionality into OpenSCAD itself. Constantly having to upload files created in one software (OpenSCAD) to a website in order to customize your models can get a little problematic as one is uploading scripts without being able to confirm how the script will translate into a form on the site. Wouldnt it be great if everything is at one place, the original place: OpenSCAD? Of course, it would.

My project intends to define a user interface to customize models interactively instead of having to modify them manually. It will enable the user to create the templates for a given model which can further be changed as per users requirements.

This project will allow the modelers to create generic models (templates) which others can then customize to cater to their own use.

This project is based on the User interface of OpenSCAD Software. The main idea of this project is to provide users with features to change certain variables or parameters in .scad file using form like interface which may include slide bar, check box, text box, ranges etc. so that we can visualize the changes in output on the basis of input side by side instead of manually changing different parameters. It will help the user able to create the templates for given model which can further be changed as per user's requirements.

The core part of this project is implemented using C++, Flex and bison and for GUI part Qt is used. Apart for that Make, Qmake and Cmake are used for making project and test cases. Testing is done using the Travis and cTest. Github is used to manage code and IRC to communicate with the mentors.

My training being not based on particular language or technology, different type of open-source software's and technologies are used in this project and many during my training which are not used in this project like Android, Ejabberd (server based on erlang and XMPP protocol) for *sunehaG* and GRASS GIS, Python, Shell Scripting for *The Road Project*.

2.2 The Existing System

The only System exiting to solve above problem is Thingiverse's Customizer which allows you to design parametric objects that can be customized with an easy web interface. They currently support OpenSCAD designs. Just upload your OpenSCAD script to Thingiverse and then anyone can open it in Customizer and customize it. Also, if you tag your Thing with the "customizer" tag, then your Thing page will automatically display an Open In Customizer button as a shortcut.

But their syntax is little messy as they use comments to generate customizer which also limits the usability of the customizer.

Limitations of previous system

- *Not supported by OpenSCAD offically.*
- Doesn't work in combination with OpenSCAD software
- Works only in online mode
- No Command-line Support
- Based upon string processing, not on lexical analysis.
- No internal support.
- Complex workflow.
- Scripts must have all the code they need in a single .scad file

- You could only put one .scad file in your Thingiverse entry

2.3 User Requirement Analysis

User Requirements Analysis for a software system is a complete description of the requirements of the User. It includes functional Requirements and Non-functional Requirements. Non-functional requirements are requirements which impose constraints on the design or implementation.

Users of the System:

1. Modeler: Modelers are the people how the code to create model theirs for their own use or for commercial use.
2. Client: These are the people which will customize model to their need made by modelers before ordering or printing model themselves.
3. Other Softwares: Many web Apps or other software using OpenSCAD as there Backend might need to interact with the customizer.

2.3.1 Functional Requirements

- **Syntax to generate a widget to modify parameter**: This means there should be a syntax which could be used to generate a different widget for a different type of parameters. The widgets that we intend to support are:
    1. Spinbox
        (a) With Increment Size
        (b) Without Increment Size
    2. Checkbox
    3. Slider
        (a) With Increment Size
        (b) With Default Increment Size
    4. Textbox
    5. VectorWidget
    6. Combo Box
        (a) Simple
        (b) Labeled
- **Syntax to Describe parameter** This means there should be a syntax which could be used to provide a description for the parameter.
- **Syntax to Group Parameter** This means there should be a syntax which could be used to group the parameters into different groups or tab to easily manage Customizer and make customizer interface little simple.
- **Syntax to Hide parameters** This means there should be a syntax which could be used to Hiding certain parameters.

Table 2.1: Table to show the required support of widgets for different DataType

	Type of Widgets				
Type of Data	Number	String	BOOL	Vector	Range
SpinBox	Y	-	-	-	-
ComboBox	Y	Y	-	-	-
Text	Y	Y	-	O	-
Slider	Y	-	-	-	-
VectorWidget	-	-	-	Y	-
Checkbox	-	-	Y	-	-

- **Syntax to make certain parameters Global** This means there should be a syntax which could be used to make certain parameters global i.e. they are present in each and every group.
- **Save the set of parameters in JSON file** This feature means there should be a way a way that gives user the ability to save the value of the parameter and also we can apply them through the cmd-line and get the output.
- **Provides option to reset parameters** All parameters in customizer could be reset to default value just by the click of a button.
- **GUI to add Set of Parameters** This means there should be a way to store different set of parameters which represent different models from generic model.
- **Cmd-line support to apply Set of Parameters** This means that values of the parameter for given set can be applied to model using the cmd-line argument.

2.3.2 Non-functional requirements

1. Extensible: It should be able to support future functional requirements
2. Usability: Simple user interfaces that a layman can understand.
3. Modular Structure: The software should have structure. So, that different parts of software would be changed without affecting other parts.
4. Backward compatibility: Addition of new syntax should not forbid script to work correctly on the backward versions of OpenSCAD.

2.4 Feasibility Study

Feasibility study aims to uncover the strengths and weaknesses of a project. In its simplest term, the two criteria to judge feasibility are cost required and value to be attained. As such, a well-designed feasibility analysis should provide a historical background of the project, description of the project or service, details of the operations and management and legal requirements. Generally, feasibility analysis precedes technical development and project implementation. These are some feasibility factors by which we can determine that the project is feasible or not:

- **Technical feasibility**: Technological feasibility is carried out to determine whether the project has the capability, in terms of software, hardware, personnel to handle and fulfill the user requirements. This whole project is based on Open Source Environment and is part of an open source software which would be deployed on any OS.
- **Economic feasibility**: In Economic feasibility, we determine whether the benefit is gain according to the cost invested to develop the project or not. If benefits outweigh costs, only then the decision is made to design and implement the system. It is important to identify cost and benefit factors, which can be categorized as follows:

  1. Development costs.
  2. Operating costs.

  OpenSCAD's customizer is also Economically feasible with as It could be developed and maintain with zero cost as It is supported by Open source community. Plus This project is started with no intention of having any economic gain but still there is an option for donations.

2.5 Objective of Project

One of the primary benefits of OpenSCAD is the ability to design customizable content. These are designs which are parametrized using parameters or top-level variables.

Some projects utilize OpenSCAD's ability to customize designs as part of their web services. e.g.

1. Thingiverse Customizer
2. Sculpteo Parametric Designs
3. e-NABLE Handomatic.

The goal of this project is two-fold:

1. Offer an auto-generated GUI associated with a customizable design, making it easier to both create and use such designs
2. Offer an authoritative standard for how to specify meta-data to guide the generation of such a GUI.

As a temporary measure, we're also planning to support the meta-data syntax used by Thingiverse, making it possible to use the thousands of customizable designs published there.

CHAPTER 3 PROJECT DESIGN

3.1 Product Perspective

This product is supposed to be part of an open source, under the GNU general Public License. It is a CAD software for programmers i.e. you code to make models. Customizer is the idea given by the user's only to fullfill their needs and It had been in demand even before it consentment. The Customizer will extent this already powerfull system by allowing the user to design the genric models and modify the parameters without the need to modify the program itself. It will also provide a way to modify the set of parameters using cmd-line.

The following are the main features that are included in OpenSCAD'customizer

1. **Cross platform support:** Offers operating support for most of the known and commercial operating systems in form of binaries and also it can be compiled on other platforms.
2. **Allows to modify parameters using GUI:** The system allows the user to easily modify the parameters of the given scad program using the GUI instead of changing it manually.
3. **Backward Compatibility:** OpenSCAD should be backward compatibility even after new features are implemented.
4. **Easy to use syntax:** The syntax for creating the input widget, groups, and description in customizer. Should be easy to use.
5. **Abiltiy to manage parameters:** It should be easy to manage the parameters by grouping them or giving them option to hide or unhide them.
6. **Feature to save a different set of parameters:** Feature to save a different set of parameters with customized name should be provided. So, that given set could be used in future.
7. **Feature to modify saved sets:** User should be able to modify,delete the already saved sets of parameters.
8. **Feature to diable Customizer:** User should be able to diable this feature totally. So, that software should work like this feature doesn't exit.

3.2 Product Functions

Functions performed by Customizer are:

- **Provides Syntax to generate a widget to modify parameter**: This means there is a syntax which could be used to generate a different widget for a different type of parameters. The widgets that we intend to support are:
    1. Spinbox
        (a) With Increment Size
        (b) Without Increment Size

2. Checkbox
3. Slider
   (a) With Increment Size
   (b) With Default Increment Size
4. Textbox
5. VectorWidget
6. Combo Box
   (a) Simple
   (b) Labeled

Table 3.1: Table to show the support of widgets for different DataType

	Type of Widgets				
Type of Data	Number	String	BOOL	Vector	Range
SpinBox	Y	-	-	-	-
ComboBox	Y	Y	-	-	-
Text	Y	Y	-	Y	-
Slider	Y	-	-	-	-
VectorWidget	-	-	-	Y	-
Checkbox	-	-	Y	-	-

- **Provide Syntax for attribute of parameter widget to be created:** Customer also provide syntax to provide additional attribute of the parameter

Table 3.2: Table to show the support of attributes for different widgets

	Type of Widgets					
	Max	Min	List of items	Step size	Label List	Default
SpinBox	-	-	-	Y	-	Y
ComboBox	-	-	Y	-	Y	Y
Text	-	-	-	-	-	Y
Slider	Y	Y	-	Y	-	Y
Vector	-	-	-	-	-	Y
Checkbox	-	-	-	-	-	Y

- **Provides Syntax to Describe parameter**: This means there is a syntax which could be used to provide a description for the parameter.
- **Provides Syntax to Group Parameter**: This means there is a syntax which could be used to group the parameters into different groups or tab to easily manage Customizer and make customizer interface little simple.
- **Provides Syntax to Hide parameters** This means there is a syntax which could be used to Hiding certain parameters.

- **Provides Syntax to make certain parameters Global** This means there is a syntax which could be used to make certain parameters global i.e. they are present in each and every group.
- **Provides option to reset parameters:** All parameters in customizer could be reset to default value just by the click of a button.
- **Provides option to Hiding description:** Description of all the parameters in customizer could be Hidden value just by the click of a button.
- **Provides Save the set of parameters in JSON file**: This feature provides a way that gives user the ability to save the value of the parameter and also we can apply them through the cmd-line and get the output.
- **Provides GUI to add Set of Parameters** This means there is a way to store a different set of parameters which represent different models from generic model.
- **Provides Cmd-line support to apply Set of Parameters:** This means that values of the parameter for given set can be applied to model using the cmd-line argument.

3.3 User Characteristics

We have identified three potential classifications of users of our system:

1. Designers: Designers are the people how the code to create model theirs for their own use or for commercial use.
2. The Client: These are the people which will customzie model to their need made by modelers before ordering or printing model themselves.
3. Developers: These are people who might want to integrate this new feature of OpenSCAD into their systems.

3.3.1 The General User

All users can be assumed to have the following characteristics:

1. Ability to read and understand English.
2. Familiarity with the operation of the basic Graphical User Interface (GUI) components of OpenSCAD.
3. Beyond the above, no further facility with computer technology can be assumed.

3.3.2 Designers

The Designer can be assumed to have the following characteristics:

1. Basic Knowledge of OpenSCAD.
2. Basic Knowledge of Creating models.
3. Basic coding skills.
4. Optional experience of cmd-line.

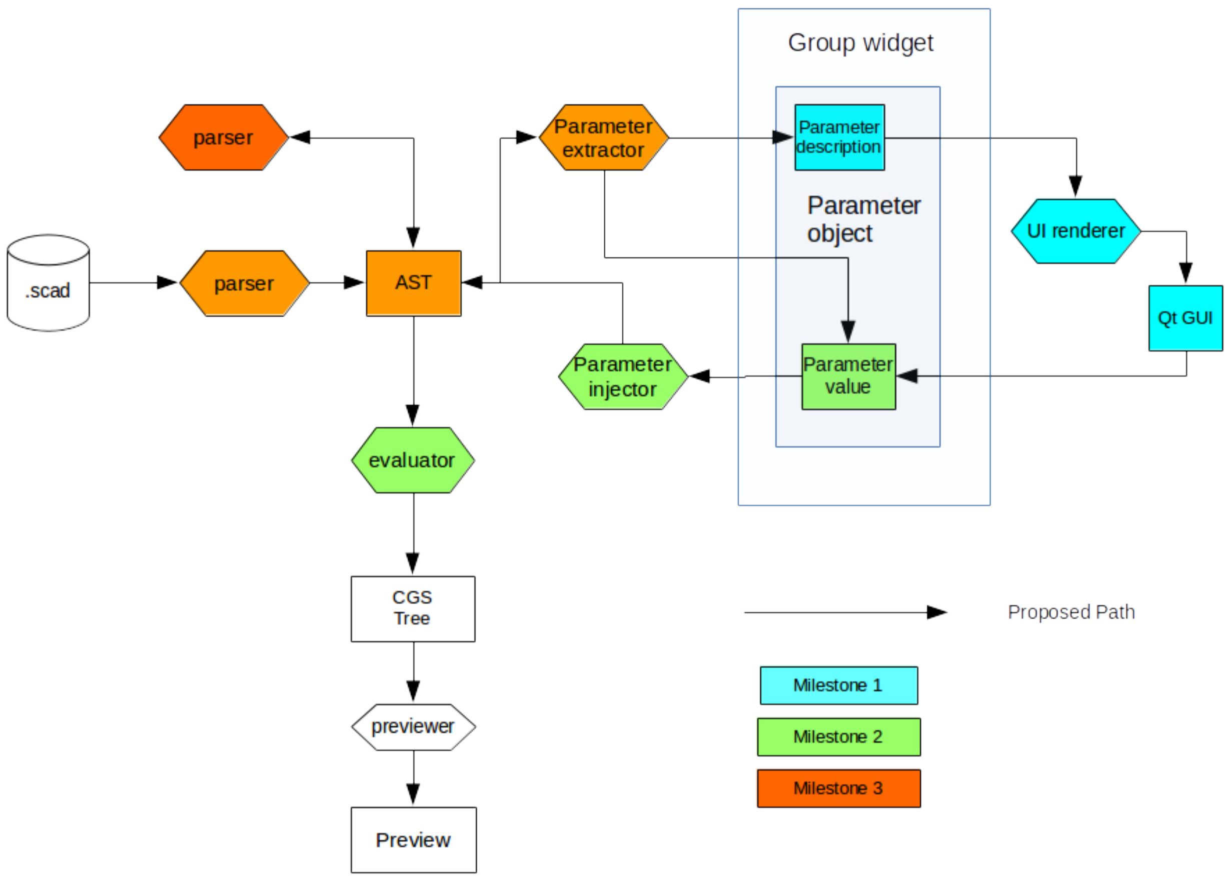


Figure 3.1: Flowchart of Customizer

3.3.3 Developers

The Developers can be assumed to have the following characteristics:

1. Basic Knowledge of programming.
2. Knowledge of JSON.
3. Ability to program in cmd-line.

3.4 Flowchart

A flowchart is a type of diagram that represents an algorithm, work flow or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows and the flowchart 3.1 of customizer showing the flow of control and Data in the software.

3.4.1 Detailed Description

The basic implementation of this project is almost done in form of prototype. There is need to modify the structure of the project. We have to divide the task into there parts:

1. **Front end** It will deal with how the parameter will look to the user like in form of range or spinbox etc. This part will include two parts:
   (a) **Individual Parameter** This will define how individual parameters will look like
   (b) **Container Widget** This will contain UI features common to all parameter. This widget will contain all parameter widget.
2. **Back End** This will include the parser part that will create AST nodes and we can extract the parameters from the AST. we can use the single parser for the whole .scad file or separate parser for extracting the parameters with annotations.

   The Back-end part will also include the parameter extractor and injector or the injector can be included in parameter object which will serve as interface
3. **Interface** This will include the parameter object which will serve as an interface between both Back end and Front end. Parameter object will contain information regarding each individual parameter like parameter name, default value, and information how this parameter will be displayed as widgets to the user. Parameter object could also include the method to inject the value of the individual parameter into the AST.

3.5 Dependency Graph

A Dependency Graph is a graphical representation of the which module is dependent on which other modules. A Dependency Graph is often used as a preliminary step to creating an overview of the system. Dependency Graph also gives overview of how good is the design of the system. OpenSCAD being were huge software it would be difficult to make the dependency graph of whole software. So, here is Dependency Graph of Customizer is as following-:

1. **Dependency graph of Comment.h:** Figure 3.4 shows the modules on which the comment.h module is dependent.
2. **Caller graph of Comment.h:** Figure 3.3 shows the modules that use the module comment.h.
3. **Caller graph of ParameterWidget.cc:** Figure 3.2 show the modules that uses the module ParameterWidget.h.

3.6 Class Diagrams

Class Diagrams describe the static structure of the system. Following classes diagram represent the relationship between different classes in OpenSCAD and customizer:

1. Figure 3.5 shows the class diagram of the ParameterWidgetVirtual class which is the basse class of all the Widget classes i.e ParameterCheckbox, ParameterVector,ParmeterSpinbox, ParameterComboBox,ParameterSlider, ParameterText.

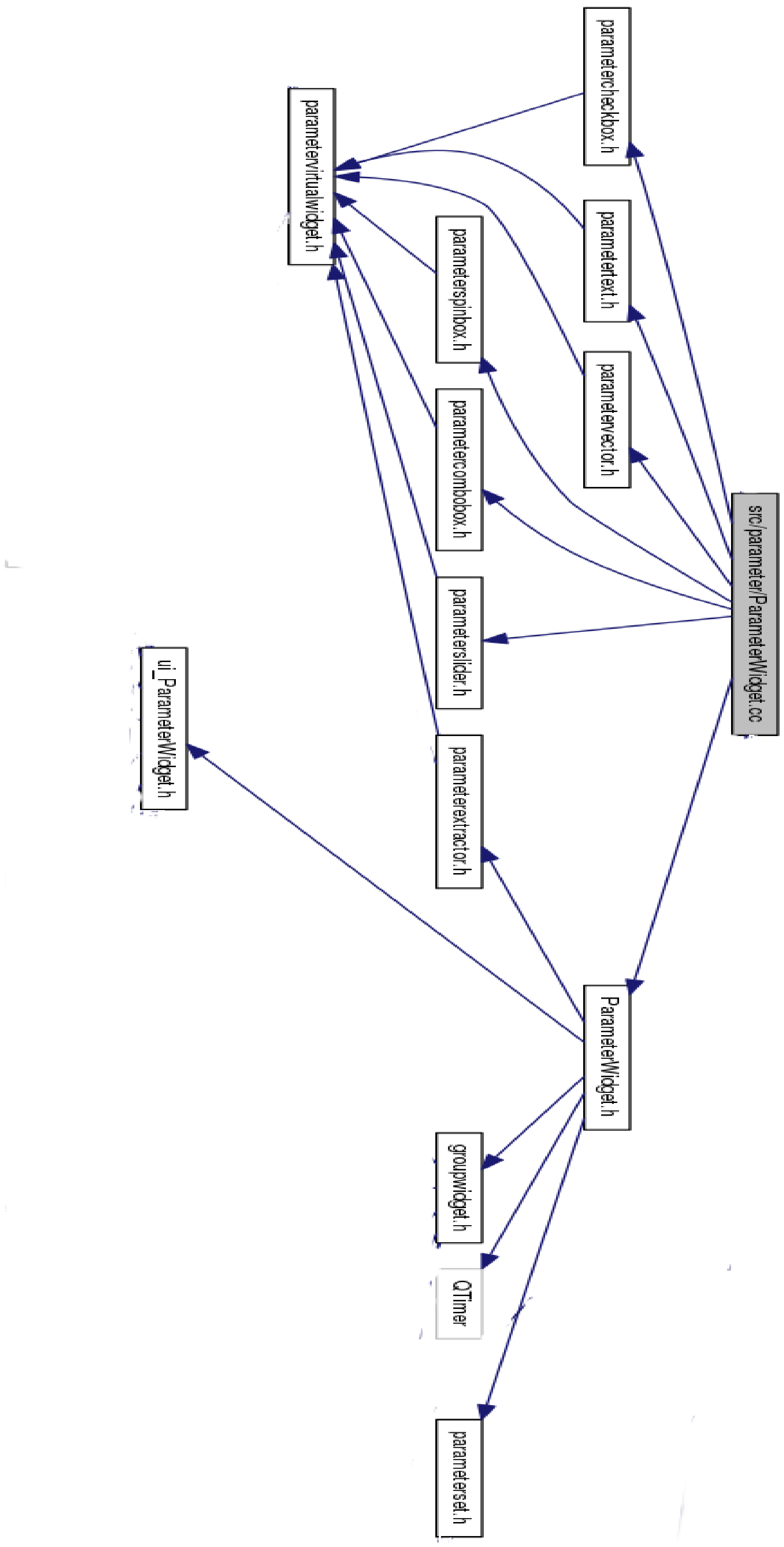


Figure 3.2: Dependency graph of Comment.h

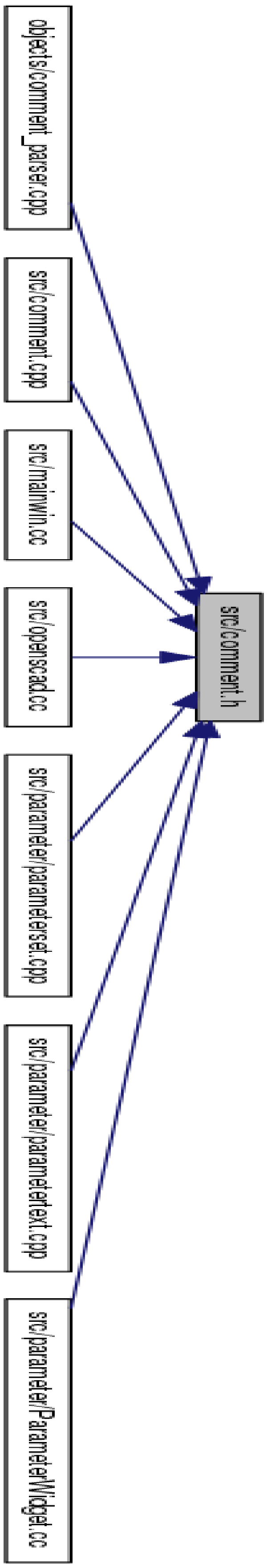

objects/comment_parser.cpp
src/comment.cpp
src/mainwin.cc
src/openscad.cc
src/parameter/parameterset.cpp
src/parameter/parametertext.cpp
src/parameter/ParameterWidget.cc
src/comment.h

Figure 3.3: Caller graph of Comment.h

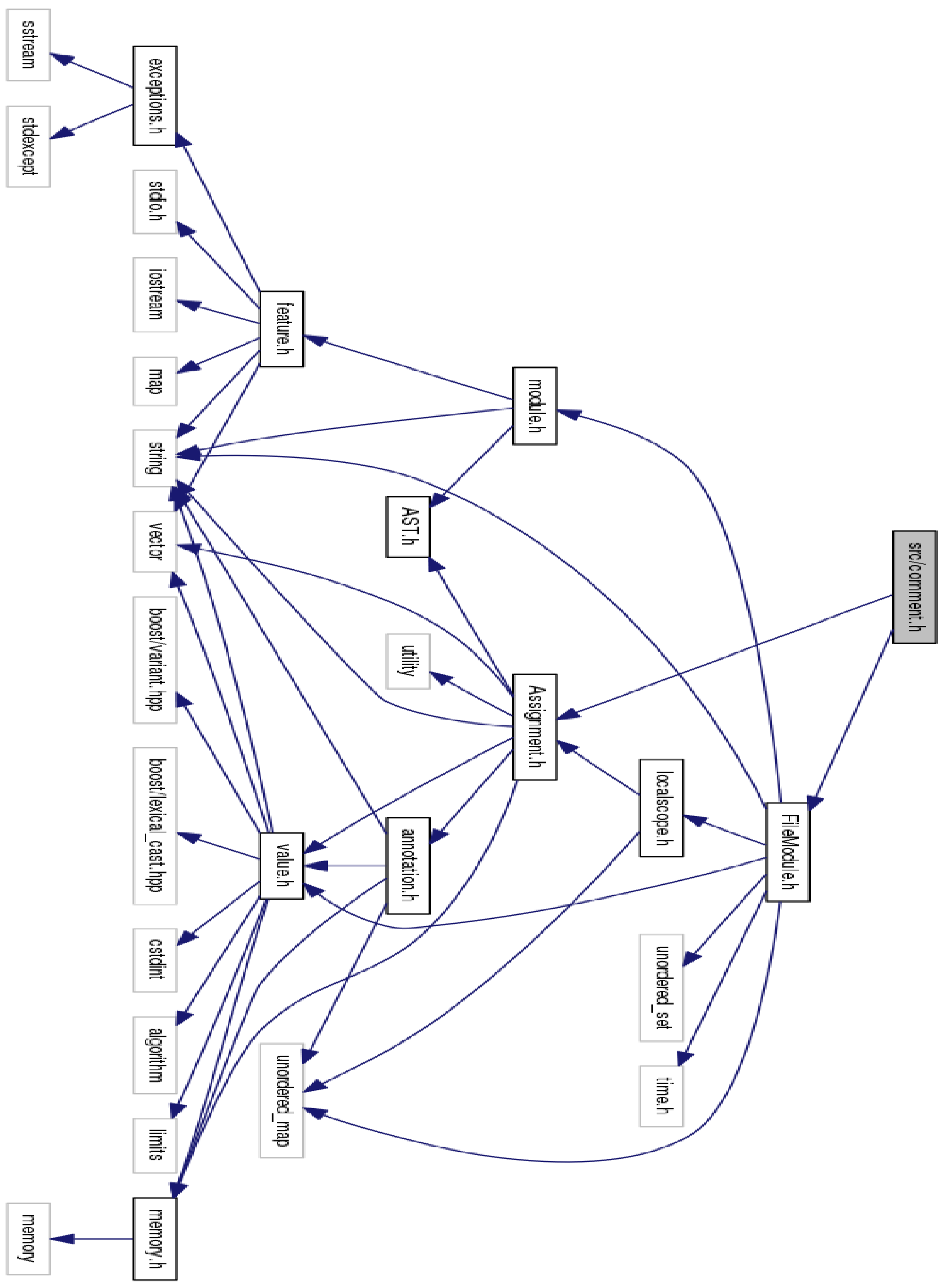


Figure 3.4: Caller graph of ParameterWidget.cc

2. Figure 3.6 shows the class diagram of the ParameterWidget class which is the main class for whole the Customizer GUI and also show how this class is related to various classes in custmoizer.

3. Figure 3.7 shows the class diagram of the Annotation class which is the main node of the AST that support the customizer feature and also show how this class is related to various classes in the customizer.

3.7 Dependencies

Dependencies include softwares or framework that need to be installed for proper working of this software.

There is no software dependencies to Install this software.

This software could be installed on any given list of operating system.

1. Mac OS X
2. Windows
   (a) XP or newer on x86 32/64 bit
3. Linux
   (a) Debian
   (b) Ubuntu
   (c) Kubuntu
   (d) Arch Linux
   (e) openSUSE
   (f) Fedora
4. BSD
   (a) NetBSD $\geq$ 6.1
   (b) FreeBSD $\geq$ 10
   (c) OpenBSD

But If you want to build OpenSCAD this software from soucre code on *any OS*, you need some libraries and tools. The version numbers in brackets specify the versions which have been used for development. Other versions may or may not work as well.

If you're using a newer version of Ubuntu, you can install these libraries from aptitude. If you're using Mac, or an older Linux/BSD, there are build scripts that download and compile the libraries from source. Follow the instructions for the platform you're compiling on below.

1. A C++ compiler supporting C++11
2. Qt4.4 $\rightarrow$ 5.x (http://qt.io/)
3. QScintilla2 (2.7 $\rightarrow$)(http://www.riverbankcomputing.co.uk/software/qscintilla/)

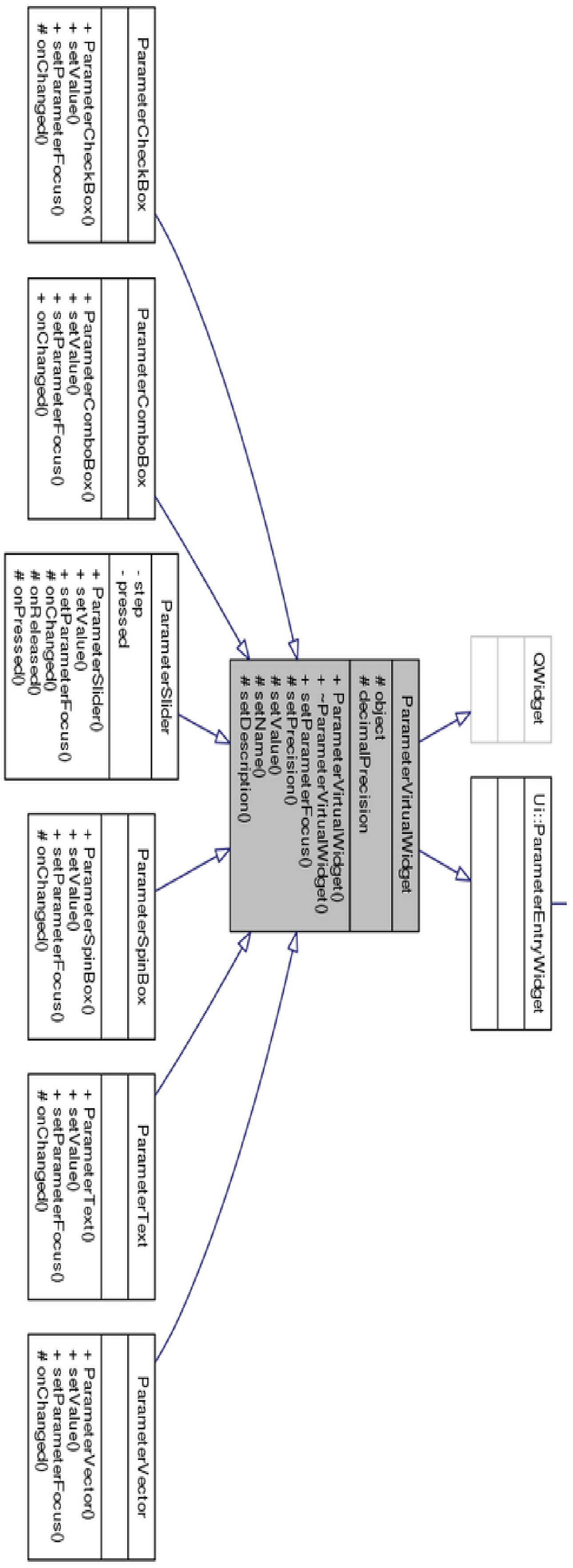


Figure 3.5: Class Diagram for Customizer (Part A)

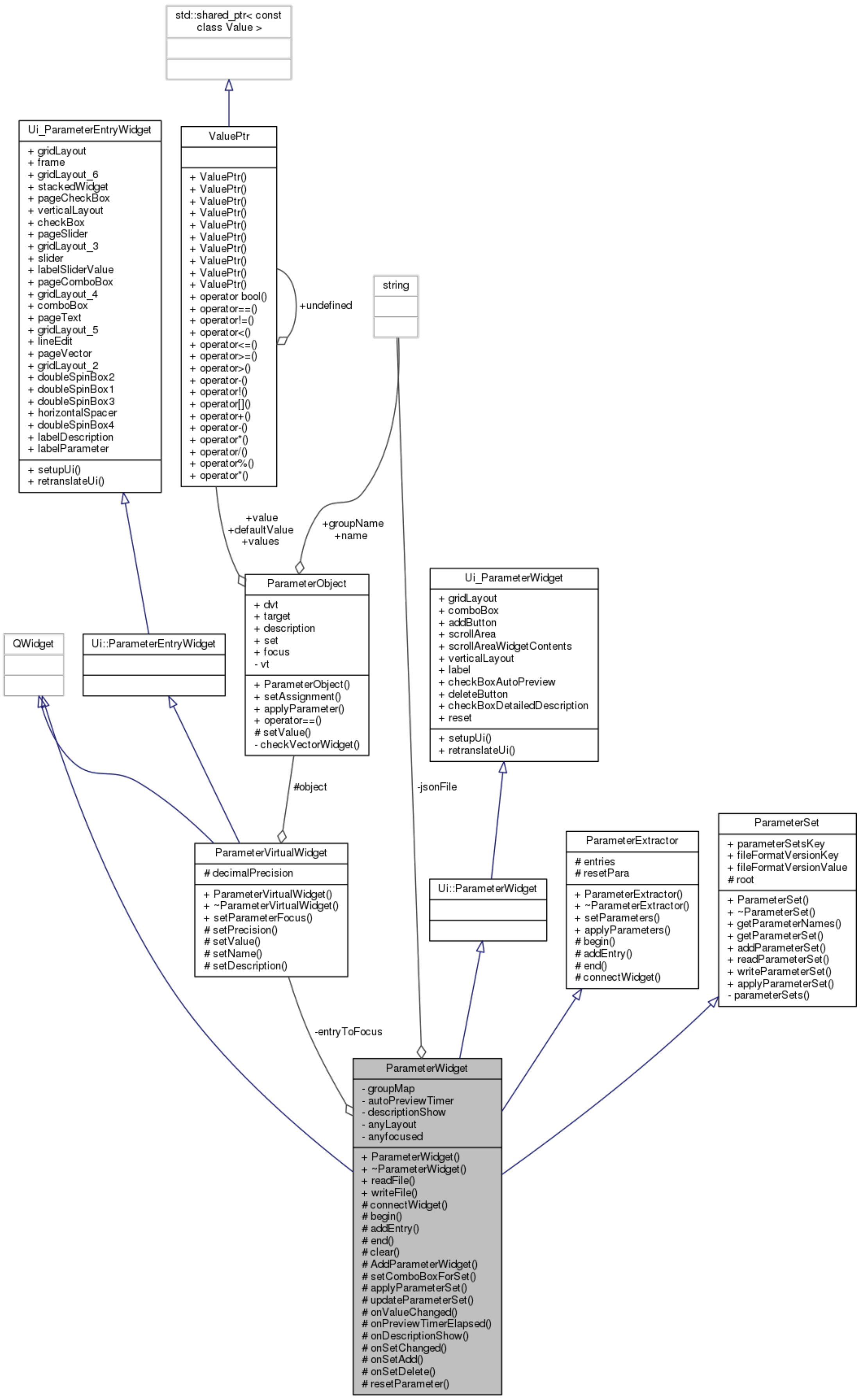


Figure 3.6: Class Diagram for Customizer (Part B)

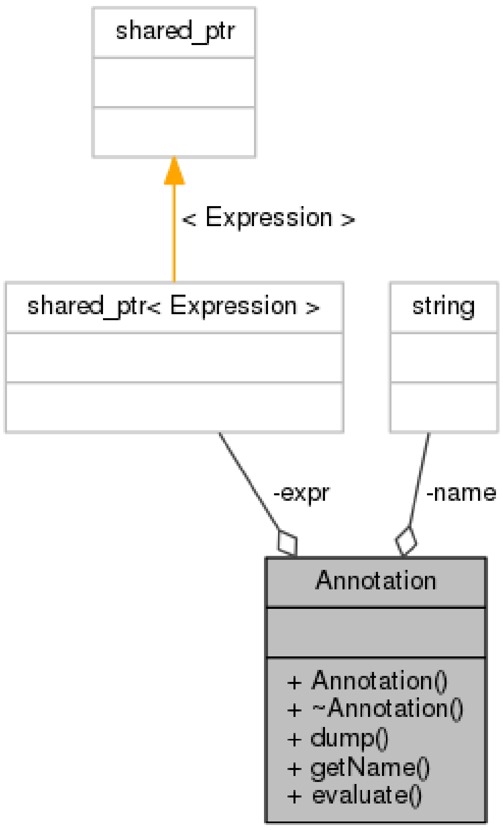


Figure 3.7: Class Diagram for Annotation

4. CGAL (3.6 →) (http://www.cgal.org/)
5. GMP (5.x) (http://www.gmplib.org/)
6. MPFR (3.x)(http://www.mpfr.org/)
7. cmake (2.8 →, required by CGAL and the test framework))(http://www.cmake.org/)
8. boost (1.35 →) (http://www.boost.org/)
9. OpenCSG (1.3.2 →)(http://www.opencsg.org/)
10. GLEW (1.5.4 →)(http://glew.sourceforge.net/)
11. Eigen (3.x)(http://eigen.tuxfamily.org/)
12. glib2 (2.x)(https://developer.gnome.org/glib/)
13. fontconfig (2.10 →)(http://fontconfig.org/)
14. freetype2 (2.4 →)(http://freetype.org/)
15. harfbuzz (0.9.19 →)(http://harfbuzz.org/)
16. Bison (2.4 →)(http://www.gnu.org/software/bison/)
17. Flex (2.5.35 →)(http://flex.sourceforge.net/)

18. pkg-config (0.26 →)(http://www.freedesktop.org/wiki/Software/pkg-config/)

CHAPTER 4 DEVELOPMENT AND IMPLEMENTATION

4.1 C++

Figure 4.1: C++ Logo

C++ is a general-purpose programming language. It has imperative, object-oriented and generic programming features, while also providing facilities for low-level memory manipulation. It was designed with a bias toward system programming and embedded, resource-constrained and large systems, with performance, efficiency and flexibility of use as its design highlights. C++ has also been found useful in many other contexts, with key strengths being software infrastructure and resource-constrained applications, including desktop applications, servers (e.g. e-commerce, web search or SQL servers), and performance-critical applications (e.g. telephone switches or space probes). C++ is a compiled language, with implementations of it available on many platforms and provided by various organizations, including the Free Software Foundation (FSF's GCC), LLVM, Microsoft, Intel and IBM.

C++ is standardized by the International Organization for Standardization (ISO), with the latest standard version ratified and published by ISO in December 2014 as ISO/IEC 14882:2014 (informally known as C++14). The C++ programming language was initially standardized in 1998 as ISO/IEC 14882:1998, which was then amended by the C++03, ISO/IEC 14882:2003, standard. The current C++14 standard supersedes these and C++11, with new features and an enlarged standard library. Before the initial standardization in 1998, C++ was developed by Bjarne Stroustrup at Bell Labs since 1979, as an extension of the C language as he wanted an efficient and flexible language similar to C, which also provided high-level features for program organization.

Many other programming languages have been influenced by C++, including C#, D, Java, and newer versions of C (after 1998).

Features of Language:

1. Object storage
    (a) Static storage duration objects
    (b) Thread storage duration objects
    (c) Automatic storage duration objects
    (d) Dynamic storage duration objects

2. Templates
3. Objects
   (a) Encapsulation
   (b) Inheritance
4. Operators and operator overloading
5. Polymorphism
   (a) Static polymorphism
   (b) Dynamic polymorphism
      i. Inheritance
      ii. Virtual member functions
6. Lambda expressions
7. Exception handling

4.2 Flex

Flex (fast lexical analyzer generator) is a free and open-source software alternative to lex. It is a computer program that generates lexical analyzers (also known as "scanners" or "lexers"). It is frequently used as the lex implementation together with Berkeley Yacc parser generator on BSD-derived operating systems (as both lex and yacc are part of POSIX), or together with GNU bison (a version of yacc) in *BSD ports and in GNU/Linux distributions. Unlike Bison, flex is not part of the GNU Project and is not released under the GNU Public License.

Flex was written in C by Vern Paxson around 1987. He was translating a Ratfor generator, which had been led by Jef Poskanzer

Input to Lex is divided into three sections with %% dividing the sections. Flex .l specification file:

```
/*** Definition section ***/

%%

/*** Rules section ***/

%%

/*** C Code section ***/
```

4.3 Bison

Bison is a general-purpose parser generator that converts an annotated context-free grammar into a deterministic LR or generalized LR (GLR) parser employing LALR(1) parser tables.

Figure 4.2: GNU Logo

Once you are proficient with Bison, you can use it to develop a wide range of language parsers, from those used in simple desk calculators to complex programming languages.

Bison is upward compatible with Yacc: all properly-written Yacc grammars ought to work with Bison with no change. Anyone familiar with Yacc should be able to use Bison with little trouble. You need to be fluent in C or C++ programming in order to use Bison or to understand this manual. Java is also supported as an experimental feature.

Bison was written originally by Robert Corbett. Richard Stallman made it Yacc-compatible. Wilfred Hansen of Carnegie Mellon University added multi-character string literals and other features. Since then, Bison has grown more robust and evolved many other new features thanks to the hard work of a long list of volunteers.

Bison .y specification file:

```
/*** Definition section ***/

%%

/*** Rules section ***/

%%

/*** C Code section ***/
```

4.4 JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures.

4.5 Qt

Figure 4.3: Qt Logo

Qt is a cross-platform application framework that is widely used for developing application software that can be run on various software and hardware platforms with little or no change in the underlying codebase, while still being a native application with native capabilities and speed. Qt is currently being developed both by The Qt Company, a company listed on the Nasdaq Helsinki Stock Exchange and the Qt Project under open-source governance, involving individual developers and firms working to advance Qt. Qt is available with both commercial and open source GPL 2.0, GPL 3.0, and LGPL 3.0 licenses.

Qt is used mainly for developing application software with graphical user interfaces (GUIs); however, programs without a GUI can be developed, such as command-line tools and consoles for servers. An example of a non-GUI program using Qt is the Cutelyst web framework. GUI programs created with Qt can have a native-looking interface, in which case Qt is classified as a widget toolkit.

Qt uses standard C++ with extensions including signals and slots that simplify handling of events, and this helps in development of both GUI and server applications which receive their own set of event information and should process them accordingly. Qt supports many compilers, including the GCC C++ compiler and the Visual Studio suite. Qt also provides Qt Quick, that includes a declarative scripting language called QML that allows using JavaScript to provide the logic. With Qt Quick, rapid application development for mobile devices became possible, although logic can be written with native code as well to achieve the best possible performance. Qt can be used in several other programming languages via language bindings. It runs on the major desktop platforms and some of the mobile platforms. It has extensive internationalization support. Non-GUI features include SQL database access, XML parsing, JSON parsing, thread management and network support.

4.6 Introduction to LaTeX

LaTeX, I had never heard about this term before doing this project, but when I came to know about it's features, found it excellent. LaTeX (pronounced /letk/, /letx/, /ltx/, or /ltk/) is a document markup language and document preparation system for the TeX typesetting program. Within the typesetting system, its name is styled as LaTeX.

LaTeX

Figure 4.4: Logo of LaTeX

Within the typesetting system, its name is styled as LaTeX. The term LaTeX refers only to the language in which documents are written, not to the editor used to write those documents. In order to create a document in LaTeX, a .tex file must be created using some form of text editor. While most text editors can be used to create a LaTeX document, a number of editors have been created specifically for working with LaTeX.

LaTeX is most widely used by mathematicians, scientists, engineers, philosophers, linguists, economists and other scholars in academia. As a primary or intermediate format, e.g., translating DocBook and other XML-based formats to PDF, LaTeX is used because of the high quality of typesetting achievable by TeX. The typesetting system offers programmable desktop publishing features and extensive facilities for automating most aspects of typesetting and desktop publishing, including numbering and cross-referencing, tables and figures, page layout and bibliographies.

LaTeX is intended to provide a high-level language that accesses the power of TeX. LaTeX essentially comprises a collection of TeX macros and a program to process LaTeXdocuments. Because the TeX formatting commands are very low-level, it is usually much simpler for end-users to use LaTeX.

4.6.1 Typesetting

LaTeX is based on the idea that authors should be able to focus on the content of what they are writing without being distracted by its visual presentation. In preparing a LaTeX document, the author specifies the logical structure using familiar concepts such as chapter, section, table, figure, etc., and lets the LaTeX system worry about the presentation of these structures. It therefore encourages the separation of layout from content while still allowing manual typesetting adjustments where needed.

4.7 Introduction to Doxygen

doxygen

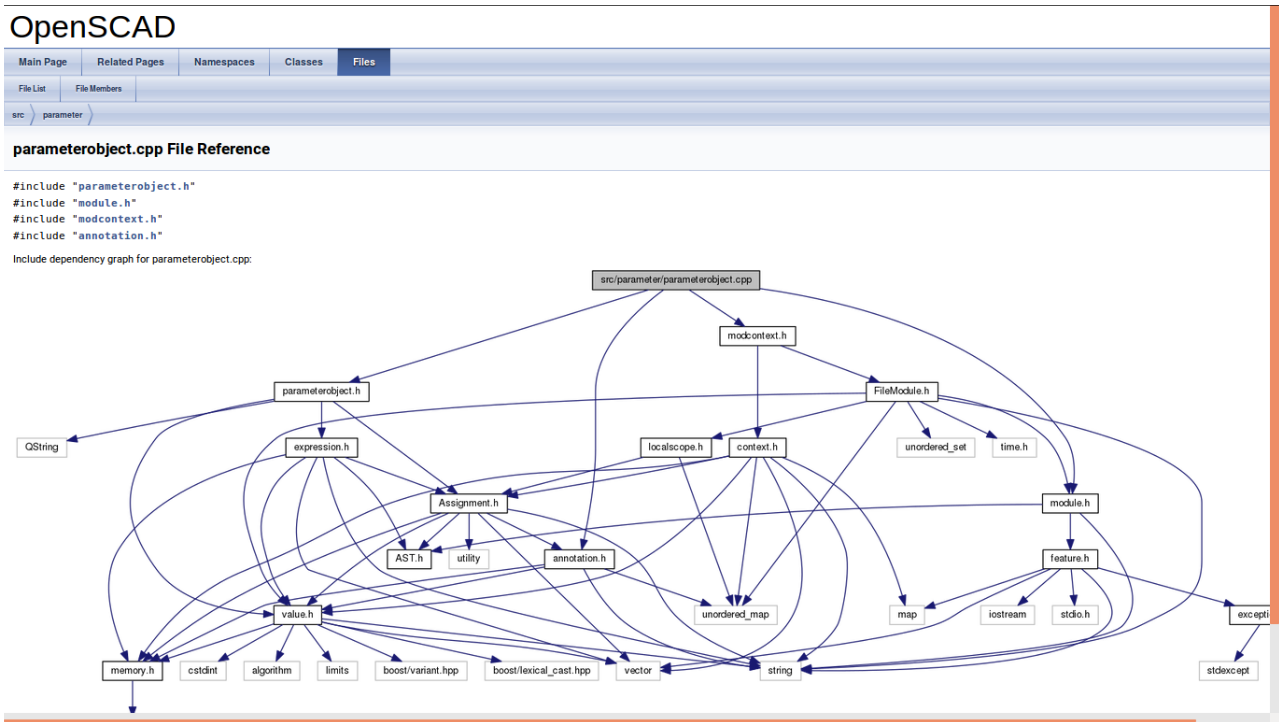
Figure 4.5: Doxygen Logo

Doxygen is a documentation generator, a tool for writing software reference documentation. The documentation is written within code, and is thus relatively easy to keep up to date. Doxygen can cross reference documentation and code, so that the reader of a document can easily refer to the actual code.

Doxygen supports multiple programming languages, especially C++, C, C#, Objective-C, Java, Python, IDL, VHDL, Fortran and PHP.[2] Doxygen is free software, released under the terms of the GNU General Public License.

4.7.1 Features of Doxygen

- Requires very little overhead from the writer of the documentation. Plain text will do, Markdown is support, and for more fancy or structured output HTML tags and/or some of doxygen's special commands can be used.
- Cross platform: Works on Windows and many Unix flavors (including Linux and Mac OS X).
- Comes with a GUI frontend (Doxywizard) to ease editing the options and run doxygen. The GUI is available on Windows, Linux, and Mac OS X.
- Automatically generates class and collaboration diagrams in HTML (as clickable image maps) and LaTeX (as Encapsulated PostScript images).
- Allows grouping of entities in modules and creating a hierarchy of modules.
- Doxygen can generate a layout which you can use and edit to change the layout of each page.
- Can cope with large projects easily.



parameterobject.cpp File Reference

```
#include "parameterobject.h"
#include "module.h"
#include "modcontext.h"
#include "annotation.h"
```

Include dependency graph for parameterobject.cpp:

Figure 4.6: Documentation using Doxygen (main page)

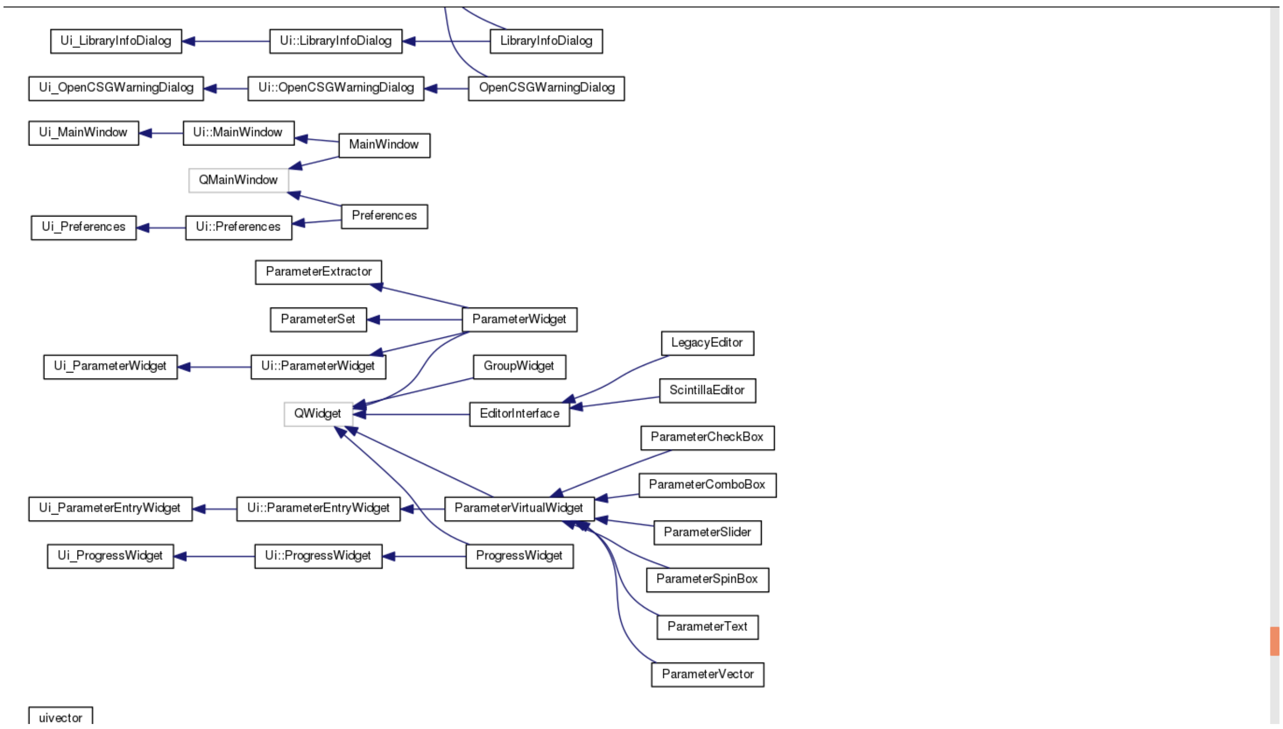
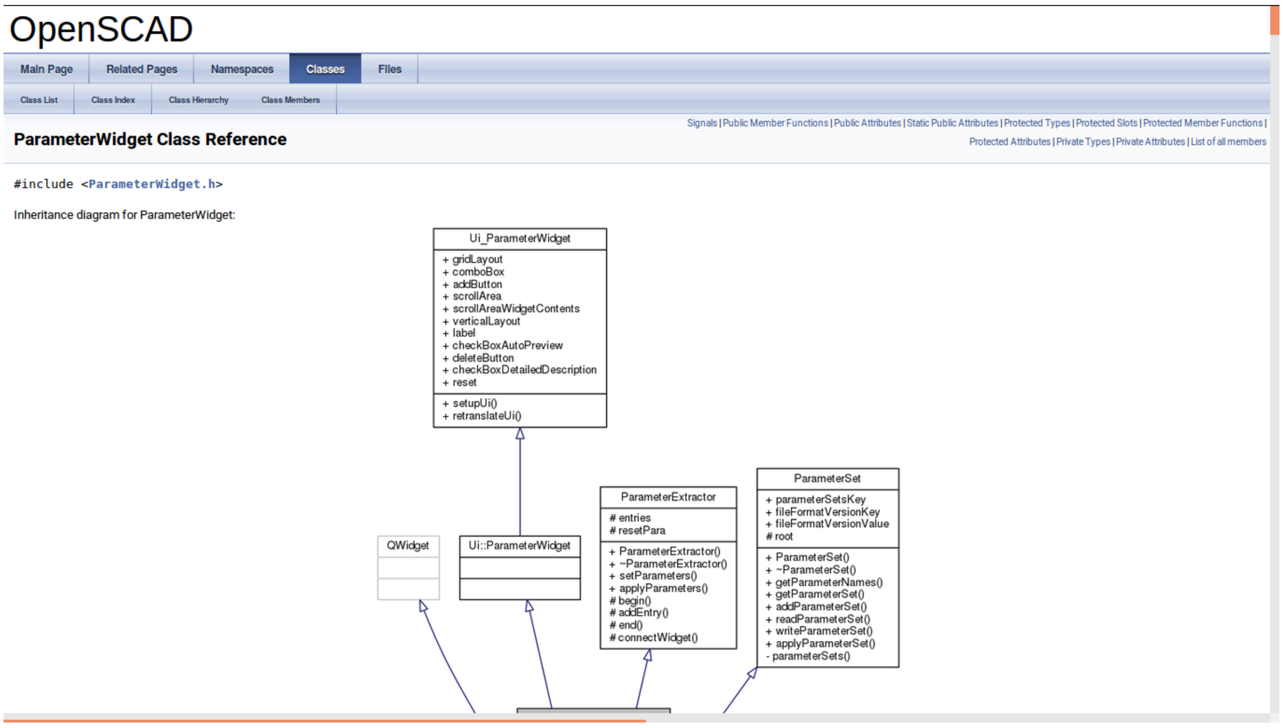


Figure 4.7: Doxygen documentation of a function


OpenSCAD
Main Page
Related Pages
Namespaces
Classes
Files
Class List
Class Index
Class Hierarchy
Class Members
Signals | Public Member Functions | Public Attributes | Static Public Attributes | Protected Types | Protected Slots | Protected Member Functions | Protected Attributes | Private Types | Private Attributes | List of all members

ParameterWidget Class Reference

```
#include <ParameterWidget.h>
```

Inheritance diagram for ParameterWidget:

Figure 4.8: Documentation using Doxygen(list of files)

4.8 Introduction to Github

GitHub is a Git repository web-based hosting service which offers all of the functionality of Git as well as adding many of its own features. Unlike Git which is strictly a command-line tool, Github provides a web-based graphical interface and desktop as well as mobile integration. It also provides access control and several collaboration features such as wikis, task management, and bug tracking and feature requests for every project.

This means that you can do things like:

- Frictionless Context Switching.
  Create a branch to try out an idea, commit a few times, switch back to where you branched from, apply a patch, switch back to where you are experimenting, and merge it in.

- Role-Based Code lines.
  Have a branch that always contains only what goes to production, another that you merge work into for testing, and several smaller ones for day to day work.

- Feature Based Work flow.
  Create new branches for each new feature you're working on so you can seamlessly switch back and forth between them, then delete each branch when that feature gets merged into your main line.

- Disposable Experimentation.
  Create a branch to experiment in, realize it's not going to work, and just delete it - abandoning the workwith nobody else ever seeing it (even if you've pushed other branches in the meantime).

Notably, when you push to a remote repository, you do not have to push all of your branches. You can choose to share just one of your branches, a few of them, or all of them. This tends to free people to try new ideas without worrying about having to plan how and when they are going to merge it in or share it with others.

4.8.1 What is Git?

Figure 4.9: Git Logo

Git is a distributed revision control and source code management (SCM) system with an emphasis on speed, data integrity, and support for distributed, non-linear workflows. Git was initially designed and developed by Linus Torvalds for Linux kernel development in 2005, and

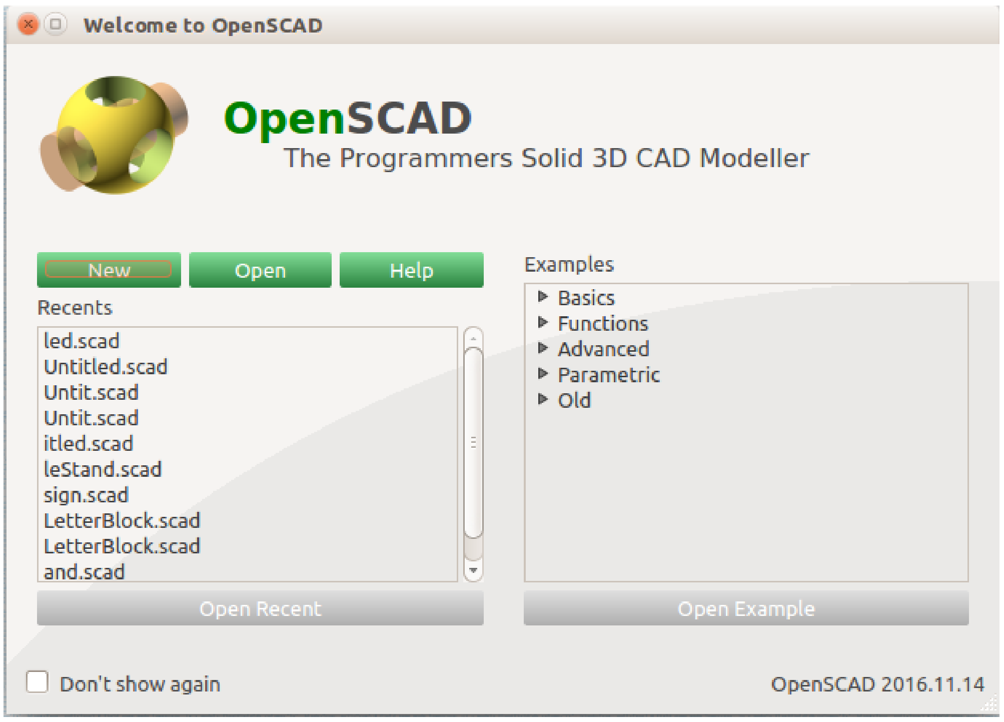


Figure 4.10: StartUp Screen for OpenSCAD

has since become the most widely adopted version control system for software development.

As with most other distributed revision control systems, and unlike most clientserver systems, every Git working directory is a full-fledged repository with complete history and full version-tracking capabilities, independent of network access or a central server. Like the Linux kernel, Git is free and open source software distributed under the terms of the GNU General Public License version 2 to handle everything from small to very large projects with speed and efficiency.

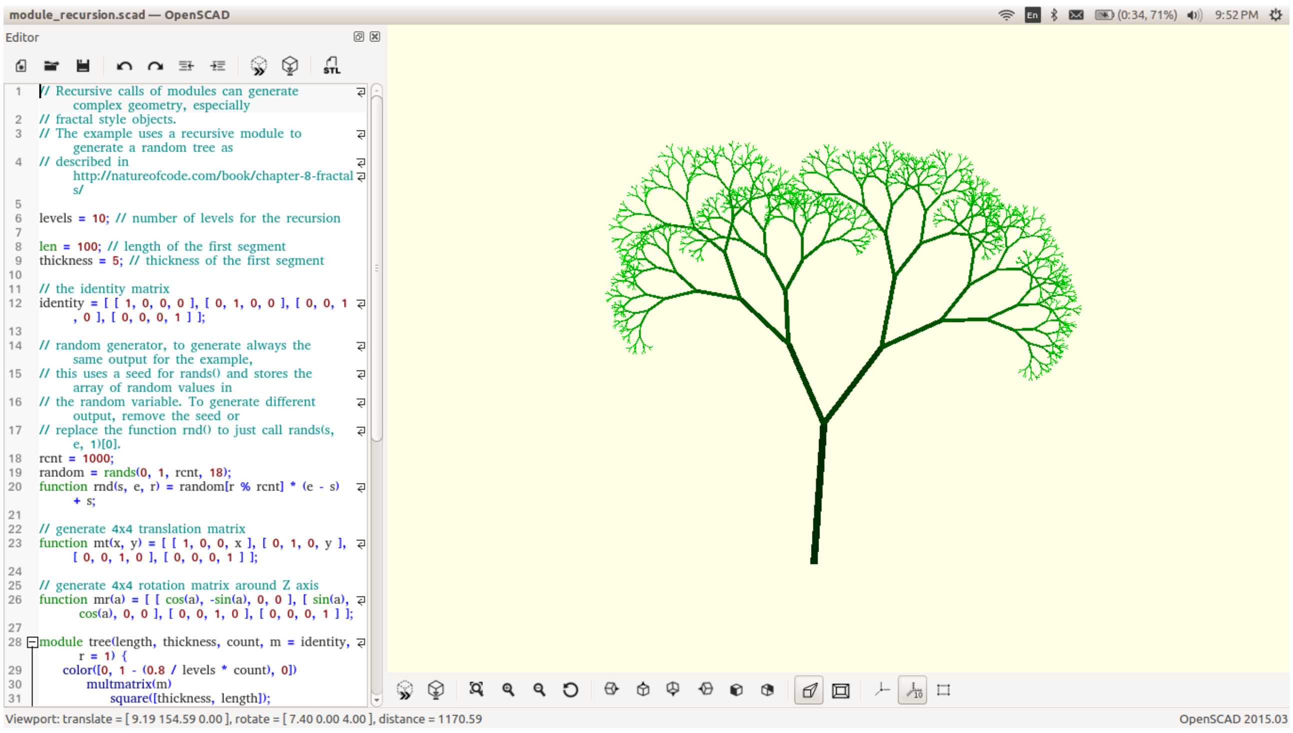
4.9 Implementation

4.9.1 Customizer

Customizer will provide User Interface to Customize Models interactively instead of modifying them manually. It will make the user able to create the templates for given model which can further be customized to cater to their need of different users and also provide a feature to save the set of parameters which define a different model using the same template of the model.

4.9.2 Activation of Customizer functions

- This is experimental functionality.So, Initially OpenSCAD will look like 4.11
- In [Edit] menu, select [Preferencews] then open tab [Features], tick Customizer, then close the window when tick shown 4.12.



```
// Recursive calls of modules can generate complex geometry, especially
// fractal style objects.
// The example uses a recursive module to generate a random tree as
// described in http://natureofcode.com/book/chapter-8-fractals/

levels = 10; // number of levels for the recursion

len = 100; // length of the first segment
thickness = 5; // thickness of the first segment

// the identity matrix
identity = [ [ 1, 0, 0, 0 ], [ 0, 1, 0, 0 ], [ 0, 0, 1, 0 ], [ 0, 0, 0, 1 ] ];

// random generator, to generate always the same output for the example,
// this uses a seed for rands() and stores the array of random values in
// the random variable. To generate different output, remove the seed or
// replace the function rnd() to just call rands(s, e, 1)[0].
rcnt = 1000;
random = rands(0, 1, rcnt, 18);
function rnd(s, e, r) = random[r % rcnt] * (e - s) + s;

// generate 4x4 translation matrix
function mt(x, y) = [ [ 1, 0, 0, x ], [ 0, 1, 0, y ], [ 0, 0, 1, 0 ], [ 0, 0, 0, 1 ] ];

// generate 4x4 rotation matrix around Z axis
function mr(a) = [ [ cos(a), -sin(a), 0, 0 ], [ sin(a), cos(a), 0, 0 ], [ 0, 0, 1, 0 ], [ 0, 0, 0, 1 ] ];

module tree(length, thickness, count, m = identity, r = 1) {
    color([0, 1 - (0.8 / levels * count), 0])
        multmatrix(m)
            square([thickness, length]);
```

Viewport: translate = [9.19 154.59 0.00], rotate = [7.40 0.00 4.00], distance = 1170.59

OpenSCAD 2015.03

Figure 4.11: OpenSCAD without customizer

- In View menu, you shall now have an option [Hide customizer], that you shall untick. Then you will be able to see the customizer 4.13

4.9.3 Syntax support for generation of the customization form

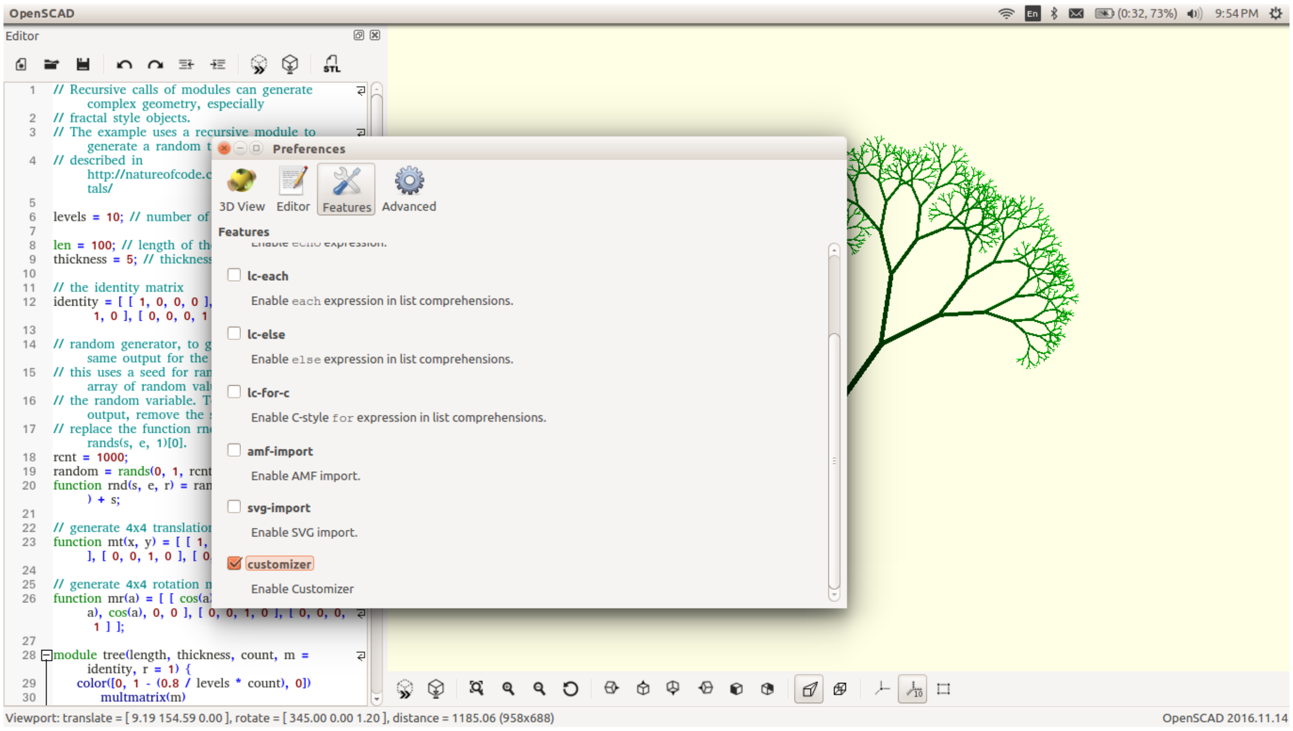
```
// variable description
variable name = defaultValue; // possible values
```

Parameter will be decorated with meta data using the comments and the single line comments above the parameter could be used to describe the meaning of the parameter and its use. The comments in same line as that of parameter is used to define the GUI that have to be used to modify that values of that parameter.

Following is the syntax for how to define different types of widgets in the form

1. **Drop down box:** 4.14 Following type of comboBox could be created with following syntax:

   ```
   // combo box for nunber
   Numbers=2; // [0, 1, 2, 3]

   // combo box for string
   Strings="foo"; // [foo, bar, baz]

   //labeled combo box for numbers
   Labeled_values=10; // [10:L, 20:M, 30:L]
   ```


OpenSCAD

Editor

```
// Recursive calls of modules can generate
    complex geometry, especially
// fractal style objects.
// The example uses a recursive module to
    generate a random t
// described in
    http://natureofcode.c
    tals/

levels = 10; // number of

len = 100; // length of th
thickness = 5; // thicknes

// the identity matrix
identity = [ [ 1, 0, 0, 0 ],
    1, 0 ], [ 0, 0, 0, 1

// random generator, to g
    same output for the
// this uses a seed for ran
    array of random val
// the random variable. T
    output, remove the s
// replace the function rn
    rands(s, e, 1)[0].
rcnt = 1000;
random = rands(0, 1, rcnt
function rnd(s, e, r) = ran
    ) + s;

// generate 4x4 translatio
function mt(x, y) = [ [ 1,
    ], [ 0, 0, 1, 0 ], [ 0

// generate 4x4 rotation m
function mr(a) = [ [ cos(a
    a), cos(a), 0, 0 ], [ 0, 0, 1, 0 ], [ 0, 0, 0,
    1 ] ];

module tree(length, thickness, count, m =
    identity, r = 1) {
  color([0, 1 - (0.8 / levels * count), 0])
    multmatrix(m)
```

Preferences

Features

Enable echo expression.

☐ lc-each
Enable each expression in list comprehensions.

☐ lc-else
Enable else expression in list comprehensions.

☐ lc-for-c
Enable C-style for expression in list comprehensions.

☐ amf-import
Enable AMF import.

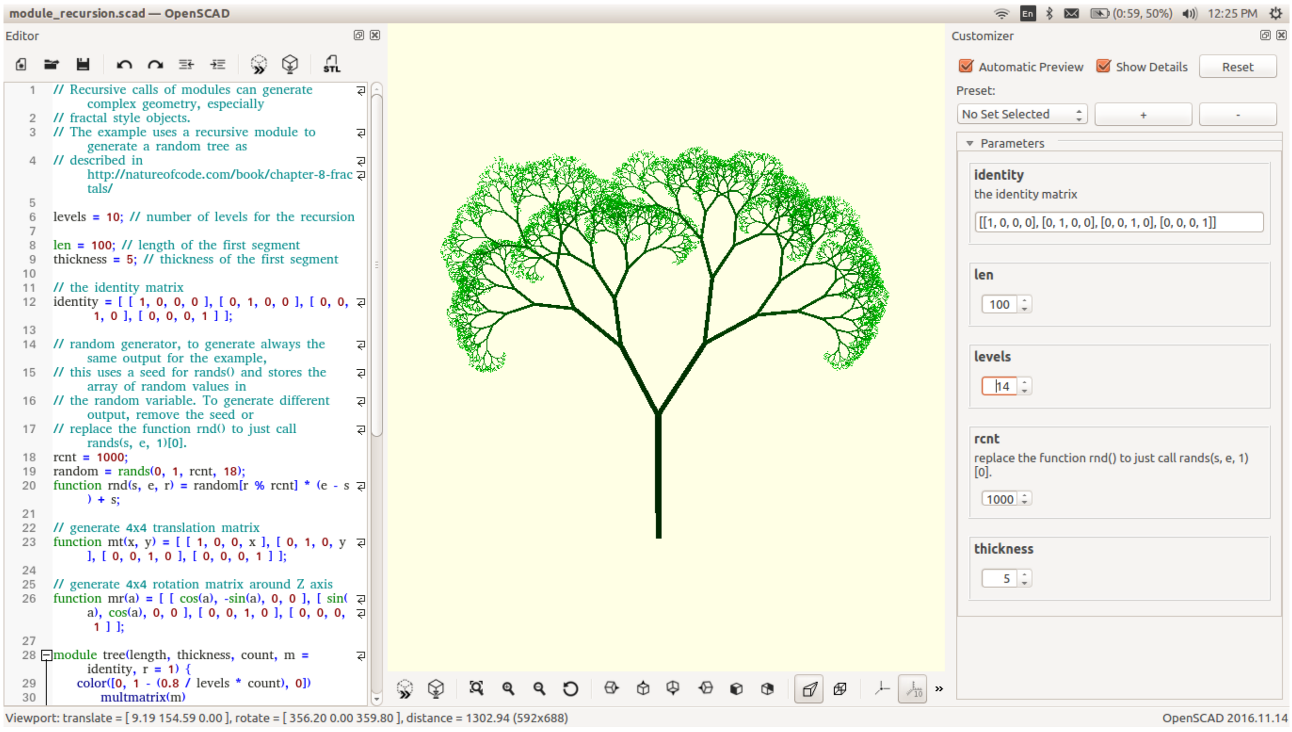
☐ svg-import
Enable SVG import.

☑ customizer
Enable Customizer

Viewport: translate = [9.19 154.59 0.00], rotate = [345.00 0.00 1.20], distance = 1185.06 (958x688)

OpenSCAD 2016.11.14

Figure 4.12: Preferences Widget to activate Customizer


module_recursion.scad — OpenSCAD

Editor

```
// Recursive calls of modules can generate
    complex geometry, especially
// fractal style objects.
// The example uses a recursive module to
    generate a random tree as
// described in
    http://natureofcode.com/book/chapter-8-frac
    tals/

levels = 10; // number of levels for the recursion

len = 100; // length of the first segment
thickness = 5; // thickness of the first segment

// the identity matrix
identity = [ [ 1, 0, 0, 0 ], [ 0, 1, 0, 0 ], [ 0, 0,
    1, 0 ], [ 0, 0, 0, 1 ] ];

// random generator, to generate always the
    same output for the example,
// this uses a seed for rands() and stores the
    array of random values in
// the random variable. To generate different
    output, remove the seed or
// replace the function rnd() to just call
    rands(s, e, 1)[0].
rcnt = 1000;
random = rands(0, 1, rcnt, 18);
function rnd(s, e, r) = random[r % rcnt] * (e - s
    ) + s;

// generate 4x4 translation matrix
function mt(x, y) = [ [ 1, 0, 0, x ], [ 0, 1, 0, y
    ], [ 0, 0, 1, 0 ], [ 0, 0, 0, 1 ] ];

// generate 4x4 rotation matrix around Z axis
function mr(a) = [ [ cos(a), -sin(a), 0, 0 ], [ sin(
    a), cos(a), 0, 0 ], [ 0, 0, 1, 0 ], [ 0, 0, 0,
    1 ] ];

module tree(length, thickness, count, m =
    identity, r = 1) {
  color([0, 1 - (0.8 / levels * count), 0])
    multmatrix(m)
```

Customizer

☑ Automatic Preview ☑ Show Details Reset

Preset: No Set Selected + -

Parameters

identity
the identity matrix
[[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]]

len
100

levels
14

rcnt
replace the function rnd() to just call rands(s, e, 1)[0].
1000

thickness
5

Viewport: translate = [9.19 154.59 0.00], rotate = [356.20 0.00 359.80], distance = 1302.94 (592x688)

OpenSCAD 2016.11.14

Figure 4.13: OpenSCAD with Customizer

Figure 4.14: Shows the different types ComboBox, Slider

Figure 4.15: Shows the CheckBox, TextBox, SpinBox, VectorWidget

```
//labeled combo box for string
Labeled_value="S"; // [S:Small, M:Medium, L:Large]
```

2. **Slider:** Only numbers are allowed in this one, 4.14 specify any of the following:

```
// slider widget for number
slider =34; // [10:100]

//step slider for number
stepSlider=2; //[0:5:100]
```

3. **Checkbox:** 4.15 Following widget is created with given syntax:

```
//description
Variable = true;
```

4. **Spinbox:** 4.15 Following widget is created with given syntax:

```
// spinbox with step size 1
Spinbox= 5;

// spinbox with step size 0.01
Spinbox= 5.11;

//spinbox with given step size
    SpinboxWithStep= 5; //2
```

5. **Textbox:** 4.15 Following widget is created with given syntax:

```
//Text box for vector with more than 4 elements
Vector=[12,34,44,43,23,23];

// Text box for string
String="hello";
```

6. **Special vector:** 4.15 Following widget is created with given syntax:

```
//Text box for vector with less than or equal to 4 elements
Vector2=[12,34,45,23];
```

4.9.4 Creating Tabs

Parameters can be grouped into **tabs**. This feature will allow us to separate similar and related parameters. The syntax for this is also mainly similar to that of Thingiverse syntax for creating the tabs. To create a tab, use a multi-line block comment like this:

/* [Tab Name] */

Screenshot number 4.16 Shows the implemenation of this feature.

The following tab names are reserved for special functionality:

Global Parameters in the global tab will always be shown on every tab no matter which tab is selected. Note: there will be no tab for Global params, they will just always be shown in all the tabs.

Hidden Parameters in the hidden tab will never be displayed. Not even the tab will be shown. Even though the variables who have not been parameterized using the Thingiverse or native syntax will not be displayed in OpenSCAD parameter widget but we have implemented this to make our comment like syntax similar as that of Thingiverse.

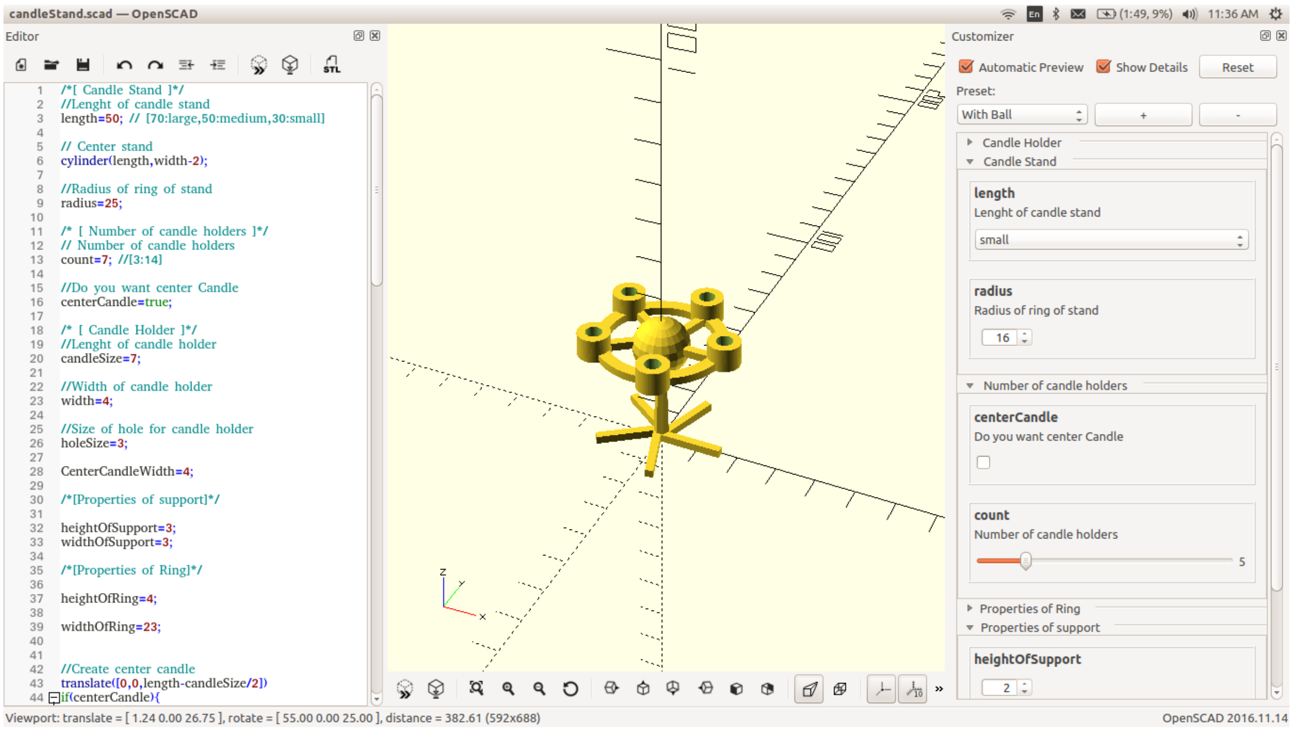
Also, the parameters who are under no tab will be displayed under TAB named parameters.

```
// combo box for nunber
Numbers=2; // [0, 1, 2, 3]

// combo box for string
Strings="foo"; // [foo, bar, baz]


/*[ Slider ]*/
// slider widget for number
slider =34; // [10:100]

//step slider for number
stepSlider=2; //[0:5:100]

/* [Global] */

//description
Variable = true;

/*[Hidden] */

// spinbox with step size 1
Spinbox = 5;

/* [Textbox] */

//Text box for vector with more than 4 elements
Vector=[12,34,44,43,23,23];

// Text box for string
String="hello";
```

4.9.5 Saving Parameters value in JSON file

This feature which is unique to openSCAD give the user the ability to save the values of all parameters and also we can apply them through the cmd-line and get the output.

And JSON file is written in the following format:


candleStand.scad — OpenSCAD
Editor
Customizer
Viewport: translate = [1.24 0.00 26.75], rotate = [55.00 0.00 25.00], distance = 382.61 (592x688)
OpenSCAD 2016.11.14

```
/*[ Candle Stand ]*/
//Lenght of candle stand
length=50; // [70:large,50:medium,30:small]

// Center stand
cylinder(length,width-2);

//Radius of ring of stand
radius=25;

/* [ Number of candle holders ]*/
// Number of candle holders
count=7; //[3:14]

//Do you want center Candle
centerCandle=true;

/* [ Candle Holder ]*/
//Lenght of candle holder
candleSize=7;

//Width of candle holder
width=4;

//Size of hole for candle holder
holeSize=3;

CenterCandleWidth=4;

/*[Properties of support]*/

heightOfSupport=3;
widthOfSupport=3;

/*[Properties of Ring]*/

heightOfRing=4;

widthOfRing=23;


//Create center candle
translate([0,0,length-candleSize/2])
if(centerCandle){
```

Automatic Preview [✓] Show Details [✓] Reset
Preset: With Ball + -
Candle Holder
Candle Stand
length
Lenght of candle stand
small
radius
Radius of ring of stand
16
Number of candle holders
centerCandle
Do you want center Candle
☐
count
Number of candle holders
5
Properties of Ring
Properties of support
heightOfSupport
2

Figure 4.16: Shows different groups generated through customzier

```
{
    "parameterSets":
    {
        "set-name":
        {
            "parameter-name"  :"value",
            "parameter-name"  :"value"
        },
        "set-name":{
            "parameter-name"  :"value",
            "parameter-name"  :"value"
        },
    },
    "fileFormatVersion": "1"
}
```

Example:

```
{
    "parameterSets":
    {
        "FirstSet":
        {
            "Labled_values": "13",
            "Numbers": "18",
            "Spinbox": "35",
            "Vector": "[2,34,45,12,23,56]",
            "slider": "2",
            "stepSlider": "12",
            "string": "he"
        },
        "SeconSet":
        {
            "Labled_values": "10",
            "Numbers": "8",
            "Spinbox": "5",
            "Vector": "[12,34,45,12,23,56]",
            "slider": "12",
            "stepSlider": "2",
            "string": "hello"
        }
    },
    "fileFormatVersion": "1"
}
```

You can write the JSON file using the two methods:

- Manually writing the JSON file
- Using the OpenSCAD's Customizer GUI

4.9.6 Appling Parameters sets from JSON file

To select the parameter set from the JSON file and apply them on the model. We have two options:

- Cmdline
- GUI

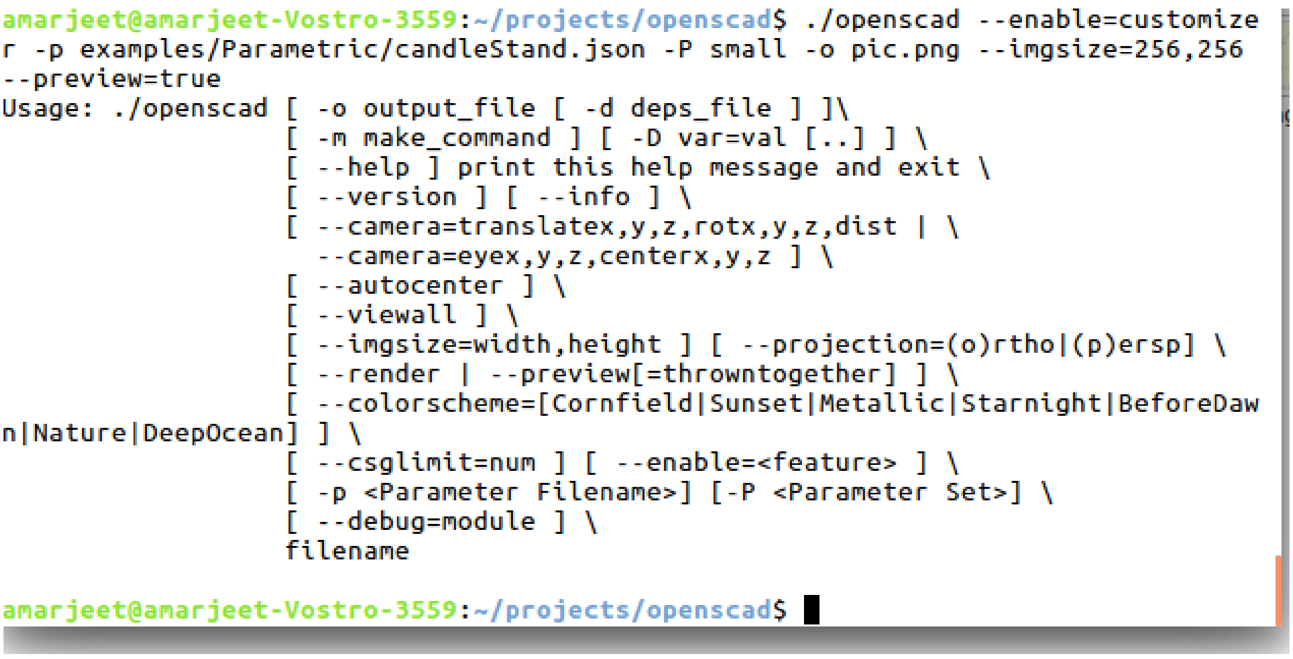
4.9.6.1 Cmdline

Cmdline option allows use for apply differenet set of parameters to the model without using the GUI and it also helps to use all the features provided by the customzier to be accessed using the cmdline. This feature will help other web-based softwares to utilize this feature. You can see various options available though cmdline in the figure 4.17

```
openscad --enable=customizer -o model-2.stl -p parameters.json -P
model-2 model.scad
```

```
openscad --enable=customizer -o <output-file> -p <parameteric-file> -P
<NameOfSet> <input-file SCAD file >
```

- -p is used to give input JSON file in which parameters are saved.
- -P is used to give the name of the set of the parameters written in JSON file.



```
amarjeet@amarjeet-Vostro-3559:~/projects/openscad$ ./openscad --enable=customize
r -p examples/Parametric/candleStand.json -P small -o pic.png --imgsize=256,256
--preview=true
Usage: ./openscad [ -o output_file [ -d deps_file ] ]\
                  [ -m make_command ] [ -D var=val [..] ] \
                  [ --help ] print this help message and exit \
                  [ --version ] [ --info ] \
                  [ --camera=translatex,y,z,rotx,y,z,dist | \
                    --camera=eyex,y,z,centerx,y,z ] \
                  [ --autocenter ] \
                  [ --viewall ] \
                  [ --imgsize=width,height ] [ --projection=(o)rtho|(p)ersp] \
                  [ --render | --preview[=throwntogether] ] \
                  [ --colorscheme=[Cornfield|Sunset|Metallic|Starnight|BeforeDaw
n|Nature|DeepOcean] ] \
                  [ --csglimit=num ] [ --enable=<feature> ] \
                  [ -p <Parameter Filename>] [-P <Parameter Set>] \
                  [ --debug=module ] \
                  filename

amarjeet@amarjeet-Vostro-3559:~/projects/openscad$
```

Figure 4.17: Cmdline options available for customizer

4.9.6.2 GUI

Through GUI you can easily apply and save Parameter in JSON file using Present section in Customizer explained below.

In customizer, You will be able to see two checkboxs which are

- **Automatic Preview:** If checked preview of the model will be automatically updated when you change any parameter in Customizer else you need to click preview button after you update parameter in the customizer.
- **Show Details:** If checked the description for the parameter will be shown above the input widget for the parameter else It will not be displayed but you still can view the description by hovering the cursor over the input widget.

Then comes **Reset** button which when clicked resets the values of all input widgets for the parameter to default provided in SCAD file.

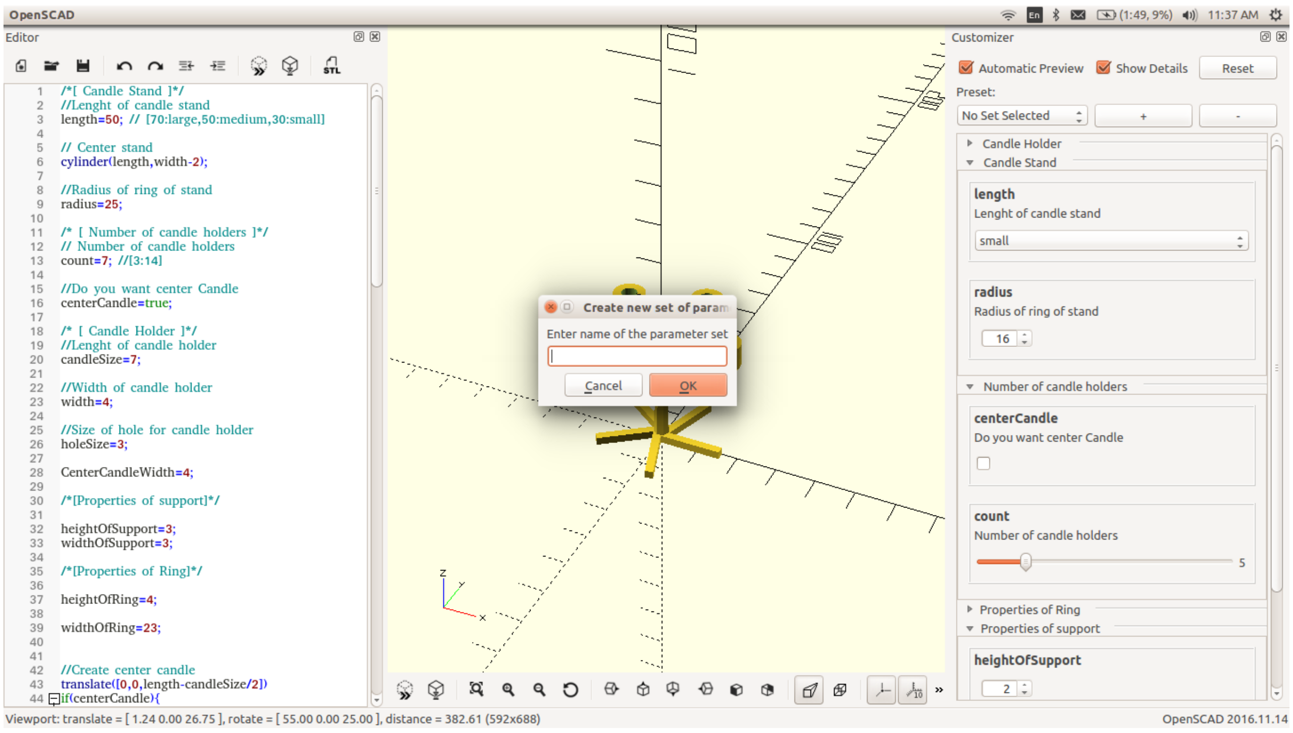
Next, come Preset section: It consist of three buttons

Combo Box: It is used to select the set of parameters to be used

+ button:

1. It is used to update the set selected in combo Box. On clicking + button values of parameters in set are replaced by new values
2. If we select No set selected in comboBox, then we can use + button to add new set of the parameters 4.18

– button: It is used to delete the set selected in combo Box. and finally below Preset Section is the Place where you can play with the parameters.



```
/*[ Candle Stand ]*/
//Lenght of candle stand
length=50; // [70:large,50:medium,30:small]

// Center stand
cylinder(length,width-2);

//Radius of ring of stand
radius=25;

/* [ Number of candle holders ]*/
// Number of candle holders
count=7; //[3:14]

//Do you want center Candle
centerCandle=true;

/* [ Candle Holder ]*/
//Lenght of candle holder
candleSize=7;

//Width of candle holder
width=4;

//Size of hole for candle holder
holeSize=3;

CenterCandleWidth=4;

/*[Properties of support]*/

heightOfSupport=3;
widthOfSupport=3;

/*[Properties of Ring]*/

heightOfRing=4;

widthOfRing=23;


//Create center candle
translate([0,0,length-candleSize/2])
if(centerCandle){
```

Create new set of param
Enter name of the parameter set

Customizer
Automatic Preview ✓ Show Details ✓ Reset
Preset: No Set Selected + -

Candle Holder
Candle Stand
length
Lenght of candle stand
small
radius
Radius of ring of stand
16
Number of candle holders
centerCandle
Do you want center Candle
☐
count
Number of candle holders
5
Properties of Ring
Properties of support
heightOfSupport
2

Viewport: translate = [1.24 0.00 26.75], rotate = [55.00 0.00 25.00], distance = 382.61 (592x688)
OpenSCAD 2016.11.14

Figure 4.18: Widget to add new set to JSON file

You can also refer to two examples that are Part of OpenSCAD to learn more

1. Parametric/sign.scad
2. Parametric/candlStand.scad

After running a set of commands of LaTeXand SageMath, it produces the output in PDF form (with pdflatex) Figure 4.16.

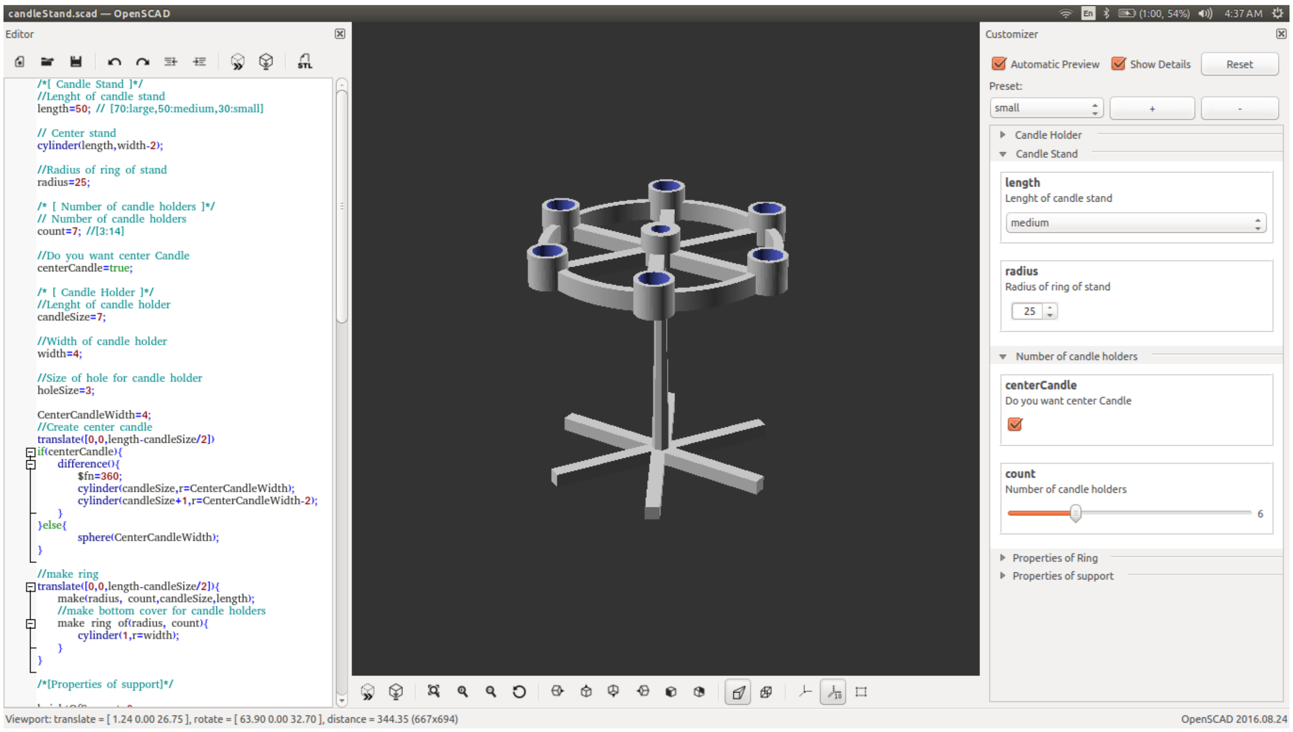
4.10 Testing

OpenSCAD is the software used by many of people all over the world and many web services as backend. so, it has to be perfect, Robust, reliable and bug-free. So, A very strong test suite is developed for testing this software and any component or feature that is to be integrated into OpenSCAD. And OpenSCAD's customizer was able to pass all the task before getting Integrated into OpenSCA.

Test Suit for OpenSCAD is developed using following two software:

1. **cTest:** CTest is a testing tool distributed as a part of CMake. It can be used to automate updating (using CVS for example), configuring, building, testing, performing memory checking, performing coverage, and submitting results to a CDash or Dart dashboard system. cTest is used by OpenSCAD for testing the OpenSCAD on the local system of developers and it performs black box and regression testing of the software.

2. **Travis CI:** Travis CI is a hosted, distributed continuous integration service used to build and test software projects hosted on GitHub.Open source projects may be tested at no



```
/*[ Candle Stand ]*/
//Lenght of candle stand
length=50; // [70:large,50:medium,30:small]

// Center stand
cylinder(length,width-2);

//Radius of ring of stand
radius=25;

/* [ Number of candle holders ]*/
// Number of candle holders
count=7; //[3:14]

//Do you want center Candle
centerCandle=true;

/* [ Candle Holder ]*/
//Lenght of candle holder
candleSize=7;

//Width of candle holder
width=4;

//Size of hole for candle holder
holeSize=3;

CenterCandleWidth=4;
//Create center candle
translate([0,0,length-candleSize/2])
if(centerCandle){
    difference(){
        $fn=360;
        cylinder(candleSize,r=CenterCandleWidth);
        cylinder(candleSize+1,r=CenterCandleWidth-2);
    }
}else{
        sphere(CenterCandleWidth);
}

//make ring
translate([0,0,length-candleSize/2]){
    make(radius, count,candleSize,length);
    //make bottom cover for candle holders
    make_ring_of(radius, count){
        cylinder(1,r=width);
    }
}

/*[Properties of support]*/
```

Customizer

Automatic Preview [✓] Show Details [✓] Reset

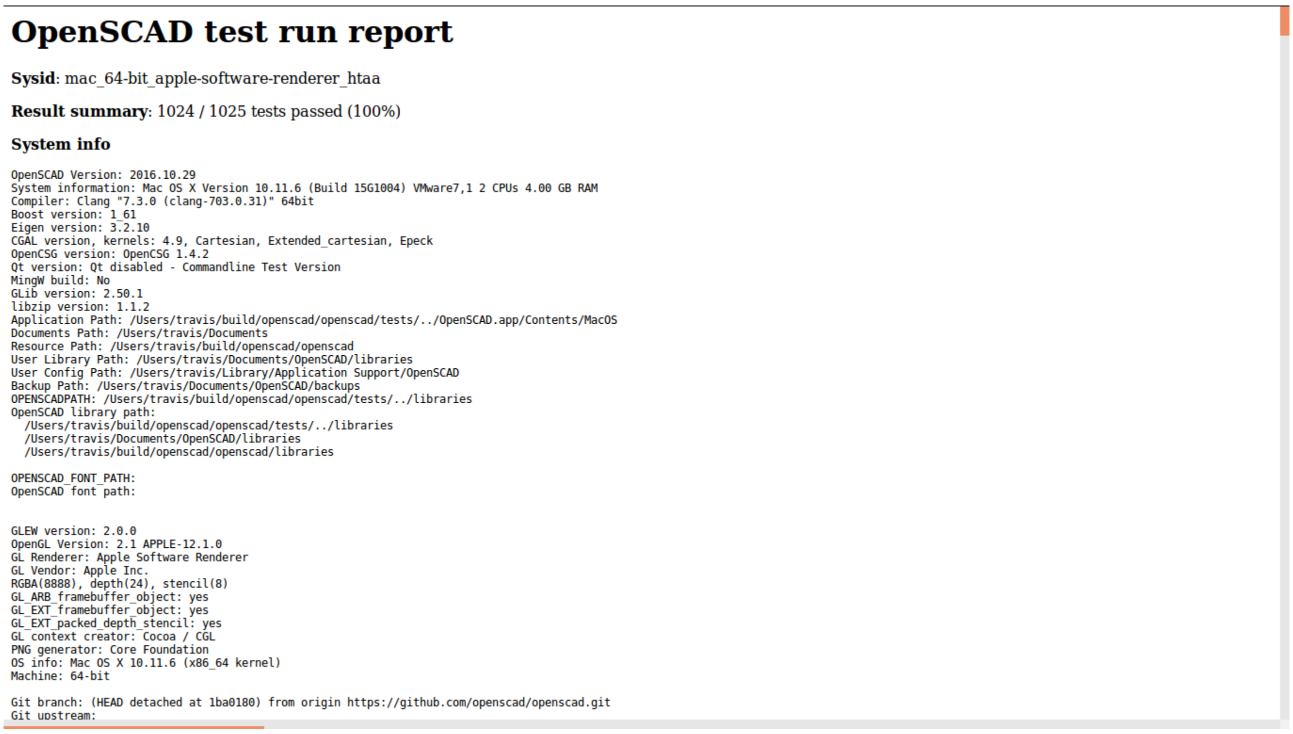
Preset: small + -

- Candle Holder
- Candle Stand
  - **length** Lenght of candle stand: medium
  - **radius** Radius of ring of stand: 25
- Number of candle holders
  - **centerCandle** Do you want center Candle: [✓]
  - **count** Number of candle holders: 6
- Properties of Ring
- Properties of support

Viewport: translate = [1.24 0.00 26.75], rotate = [63.90 0.00 32.70], distance = 344.35 (667x694)

OpenSCAD 2016.08.24

Figure 4.19: customizer with different set of parameters selected



OpenSCAD test run report

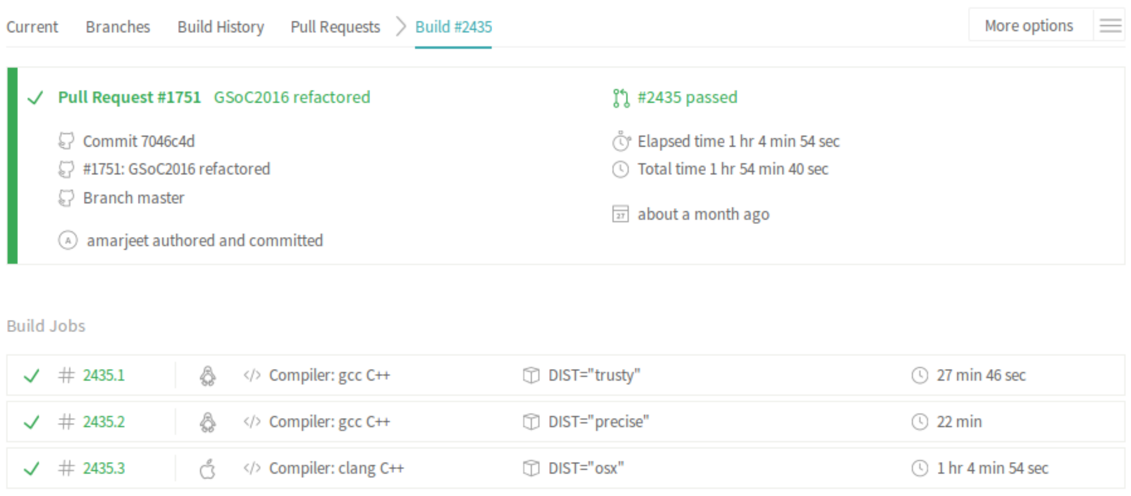
Sysid: mac_64-bit_apple-software-renderer_htaa

Result summary: 1024 / 1025 tests passed (100%)

System info

```
OpenSCAD Version: 2016.10.29
System information: Mac OS X Version 10.11.6 (Build 15G1004) VMware7,1 2 CPUs 4.00 GB RAM
Compiler: Clang "7.3.0 (clang-703.0.31)" 64bit
Boost version: 1_61
Eigen version: 3.2.10
CGAL version, kernels: 4.9, Cartesian, Extended_cartesian, Epeck
OpenCSG version: OpenCSG 1.4.2
Qt version: Qt disabled - Commandline Test Version
MingW build: No
GLib version: 2.50.1
libzip version: 1.1.2
Application Path: /Users/travis/build/openscad/openscad/tests/../OpenSCAD.app/Contents/MacOS
Documents Path: /Users/travis/Documents
Resource Path: /Users/travis/build/openscad/openscad
User Library Path: /Users/travis/Documents/OpenSCAD/libraries
User Config Path: /Users/travis/Library/Application Support/OpenSCAD
Backup Path: /Users/travis/Documents/OpenSCAD/backups
OPENSCADPATH: /Users/travis/build/openscad/openscad/tests/../libraries
OpenSCAD library path:
  /Users/travis/build/openscad/openscad/tests/../libraries
  /Users/travis/Documents/OpenSCAD/libraries
  /Users/travis/build/openscad/openscad/libraries

OPENSCAD_FONT_PATH:
OpenSCAD font path:


GLEW version: 2.0.0
OpenGL Version: 2.1 APPLE-12.1.0
GL Renderer: Apple Software Renderer
GL Vendor: Apple Inc.
RGBA(8888), depth(24), stencil(8)
GL_ARB_framebuffer_object: yes
GL_EXT_framebuffer_object: yes
GL_EXT_packed_depth_stencil: yes
GL context creator: Cocoa / CGL
PNG generator: Core Foundation
OS info: Mac OS X 10.11.6 (x86_64 kernel)
Machine: 64-bit

Git branch: (HEAD detached at 1ba0180) from origin https://github.com/openscad/openscad.git
Git upstream:
```

Figure 4.20: Test Report by travis


Current Branches Build History Pull Requests > Build #2435 More options

✓ Pull Request #1751 GSoC2016 refactored

Commit 7046c4d
#1751: GSoC2016 refactored
Branch master
amarjeet authored and committed

#2435 passed
Elapsed time 1 hr 4 min 54 sec
Total time 1 hr 54 min 40 sec
about a month ago

Build Jobs

✓	# 2435.1		Compiler: gcc C++	DIST="trusty"	27 min 46 sec
✓	# 2435.2		Compiler: gcc C++	DIST="precise"	22 min
✓	# 2435.3		Compiler: clang C++	DIST="osx"	1 hr 4 min 54 sec

Figure 4.21: Travis test summary for 3 different OS

charge via travis-ci.org. Private projects may be tested at the same location on a fee basis. Travis CI is used mainly for Installation testing on different OS and It also perform black box and regression testing on different operating systems.

Test Suit developed for OpenSCAD perform following type of testing:

1. **Installation Testing:** It is done by building the OpenSCAD from scratch on totally fresh installation of different operating systems. 4.21

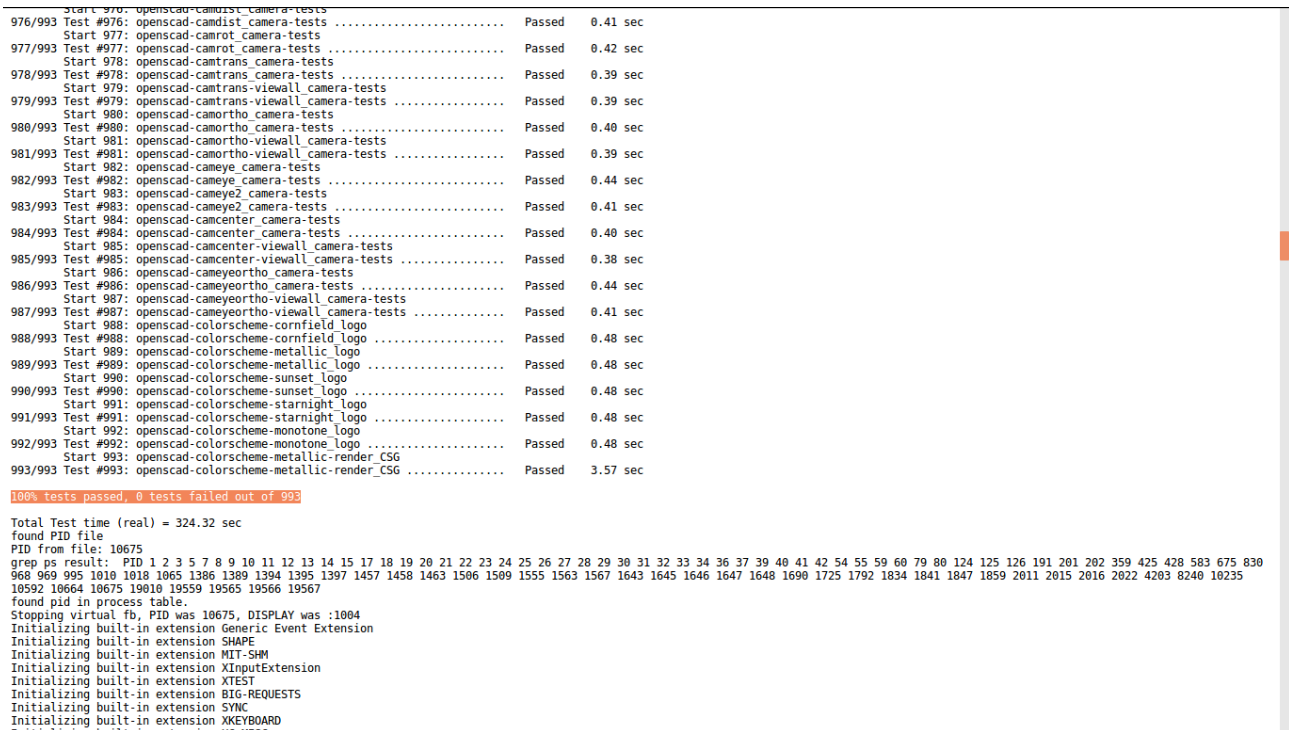
2. **Regression Testing:** It is done by running the test cases developed for the OpenSCAD before the New changes are made to check whether new Changes doesn't produce abnormal behavior. 4.22

3. **Black Box Testing:** In this Testing, the for given set of input and the output is matched with the expected output. 4.23

After passing above Test suit, OpenSCAD's customizer went through system testing, which was of two types:

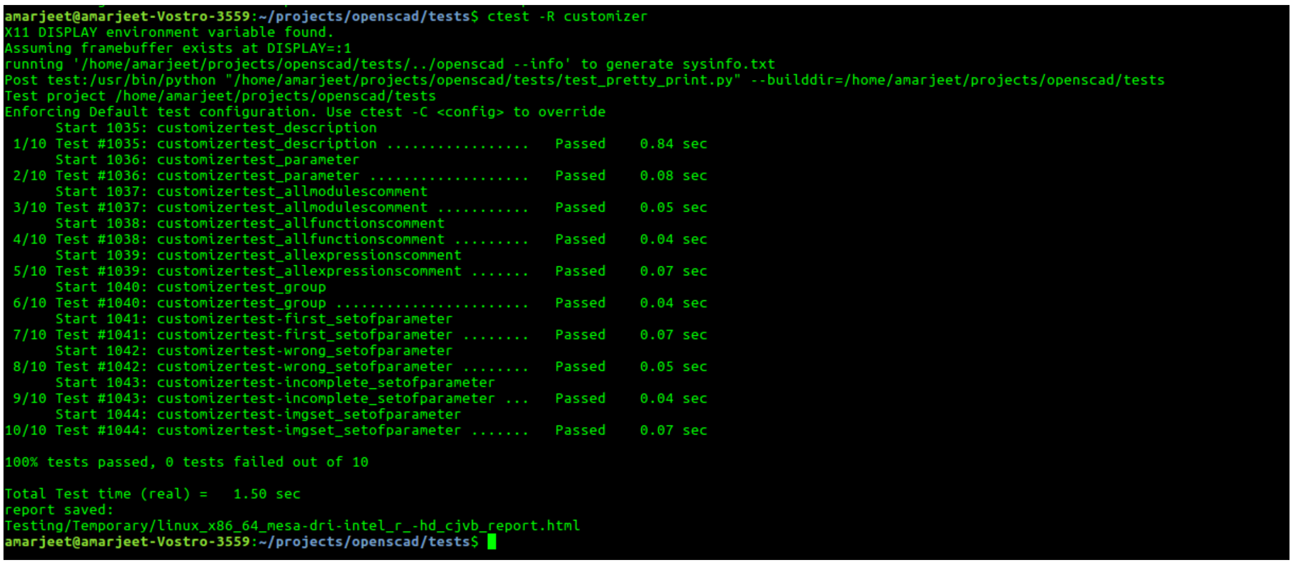
1. **Apha Testing**: In this Customizer was tested my fellow developers working for OpenSCAD community. The issues provided by them were improved and also Testing of User Interface was done. 4.25

2. **Beta Testing:** In this Customizer, was provided to community members and open to all people to test and report the issues or addition that are required to improve User Experience. 4.25



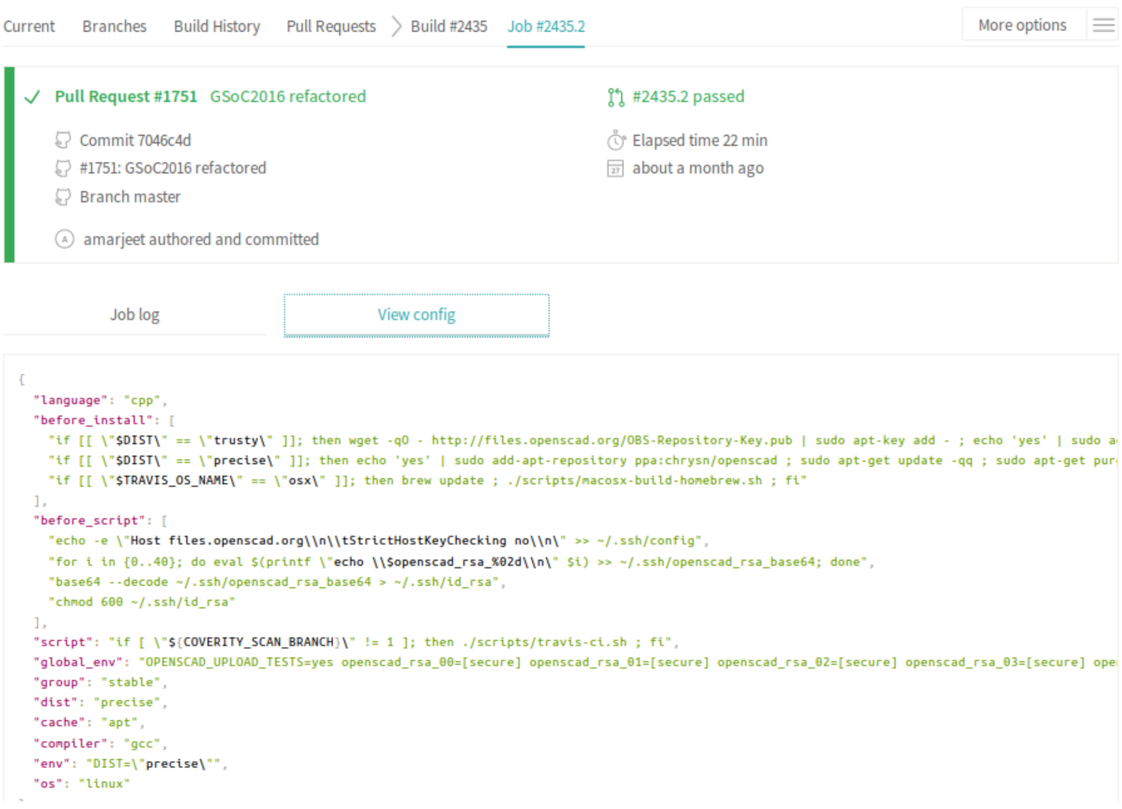
```
        Start 976: openscad-camdist_camera-tests
976/993 Test #976: openscad-camdist_camera-tests ..........................   Passed    0.41 sec
        Start 977: openscad-camrot_camera-tests
977/993 Test #977: openscad-camrot_camera-tests ...........................   Passed    0.42 sec
        Start 978: openscad-camtrans_camera-tests
978/993 Test #978: openscad-camtrans_camera-tests .........................   Passed    0.39 sec
        Start 979: openscad-camtrans-viewall_camera-tests
979/993 Test #979: openscad-camtrans-viewall_camera-tests .................   Passed    0.39 sec
        Start 980: openscad-camortho_camera-tests
980/993 Test #980: openscad-camortho_camera-tests .........................   Passed    0.40 sec
        Start 981: openscad-camortho-viewall_camera-tests
981/993 Test #981: openscad-camortho-viewall_camera-tests .................   Passed    0.39 sec
        Start 982: openscad-cameye_camera-tests
982/993 Test #982: openscad-cameye_camera-tests ...........................   Passed    0.44 sec
        Start 983: openscad-cameye2_camera-tests
983/993 Test #983: openscad-cameye2_camera-tests ..........................   Passed    0.41 sec
        Start 984: openscad-camcenter_camera-tests
984/993 Test #984: openscad-camcenter_camera-tests ........................   Passed    0.40 sec
        Start 985: openscad-camcenter-viewall_camera-tests
985/993 Test #985: openscad-camcenter-viewall_camera-tests ................   Passed    0.38 sec
        Start 986: openscad-cameyeortho_camera-tests
986/993 Test #986: openscad-cameyeortho_camera-tests ......................   Passed    0.44 sec
        Start 987: openscad-cameyeortho-viewall_camera-tests
987/993 Test #987: openscad-cameyeortho-viewall_camera-tests ..............   Passed    0.41 sec
        Start 988: openscad-colorscheme-cornfield_logo
988/993 Test #988: openscad-colorscheme-cornfield_logo ....................   Passed    0.48 sec
        Start 989: openscad-colorscheme-metallic_logo
989/993 Test #989: openscad-colorscheme-metallic_logo .....................   Passed    0.48 sec
        Start 990: openscad-colorscheme-sunset_logo
990/993 Test #990: openscad-colorscheme-sunset_logo .......................   Passed    0.48 sec
        Start 991: openscad-colorscheme-starnight_logo
991/993 Test #991: openscad-colorscheme-starnight_logo ....................   Passed    0.48 sec
        Start 992: openscad-colorscheme-monotone_logo
992/993 Test #992: openscad-colorscheme-monotone_logo .....................   Passed    0.48 sec
        Start 993: openscad-colorscheme-metallic-render_CSG
993/993 Test #993: openscad-colorscheme-metallic-render_CSG ...............   Passed    3.57 sec

100% tests passed, 0 tests failed out of 993

Total Test time (real) = 324.32 sec
found PID file
PID from file: 10675
grep ps result:  PID 1 2 3 5 7 8 9 10 11 12 13 14 15 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 36 37 39 40 41 42 54 55 59 60 79 80 124 125 126 191 201 202 359 425 428 583 675 830
968 969 995 1010 1018 1065 1386 1389 1394 1395 1397 1457 1458 1463 1506 1509 1555 1563 1567 1643 1645 1646 1647 1648 1690 1725 1792 1834 1841 1847 1859 2011 2015 2016 2022 4203 8240 10235
10592 10664 10675 19010 19559 19565 19566 19567
found pid in process table.
Stopping virtual fb, PID was 10675, DISPLAY was :1004
Initializing built-in extension Generic Event Extension
Initializing built-in extension SHAPE
Initializing built-in extension MIT-SHM
Initializing built-in extension XInputExtension
Initializing built-in extension XTEST
Initializing built-in extension BIG-REQUESTS
Initializing built-in extension SYNC
Initializing built-in extension XKEYBOARD
```

Figure 4.22: Test whole software after Integration



```
amarjeet@amarjeet-Vostro-3559:~/projects/openscad/tests$ ctest -R customizer
X11 DISPLAY environment variable found.
Assuming framebuffer exists at DISPLAY=:1
running '/home/amarjeet/projects/openscad/tests/../openscad --info' to generate sysinfo.txt
Post test:/usr/bin/python "/home/amarjeet/projects/openscad/tests/test_pretty_print.py" --builddir=/home/amarjeet/projects/openscad/tests
Test project /home/amarjeet/projects/openscad/tests
Enforcing Default test configuration. Use ctest -C <config> to override
      Start 1035: customizertest_description
 1/10 Test #1035: customizertest_description .................   Passed    0.84 sec
      Start 1036: customizertest_parameter
 2/10 Test #1036: customizertest_parameter ...................   Passed    0.08 sec
      Start 1037: customizertest_allmodulescomment
 3/10 Test #1037: customizertest_allmodulescomment ...........   Passed    0.05 sec
      Start 1038: customizertest_allfunctionscomment
 4/10 Test #1038: customizertest_allfunctionscomment .........   Passed    0.04 sec
      Start 1039: customizertest_allexpressionscomment
 5/10 Test #1039: customizertest_allexpressionscomment .......   Passed    0.07 sec
      Start 1040: customizertest_group
 6/10 Test #1040: customizertest_group .......................   Passed    0.04 sec
      Start 1041: customizertest-first_setofparameter
 7/10 Test #1041: customizertest-first_setofparameter ........   Passed    0.07 sec
      Start 1042: customizertest-wrong_setofparameter
 8/10 Test #1042: customizertest-wrong_setofparameter ........   Passed    0.05 sec
      Start 1043: customizertest-incomplete_setofparameter
 9/10 Test #1043: customizertest-incomplete_setofparameter ...   Passed    0.04 sec
      Start 1044: customizertest-imgset_setofparameter
10/10 Test #1044: customizertest-imgset_setofparameter .......   Passed    0.07 sec

100% tests passed, 0 tests failed out of 10

Total Test time (real) =   1.50 sec
report saved:
Testing/Temporary/linux_x86_64_mesa-dri-intel_r_-hd_cjvb_report.html
amarjeet@amarjeet-Vostro-3559:~/projects/openscad/tests$
```

Figure 4.23: Tests specfic to customizer



Current Branches Build History Pull Requests > Build #2435 Job #2435.2 More options

✓ Pull Request #1751 GSoC2016 refactored

Commit 7046c4d

#1751: GSoC2016 refactored

Branch master

amarjeet authored and committed

#2435.2 passed

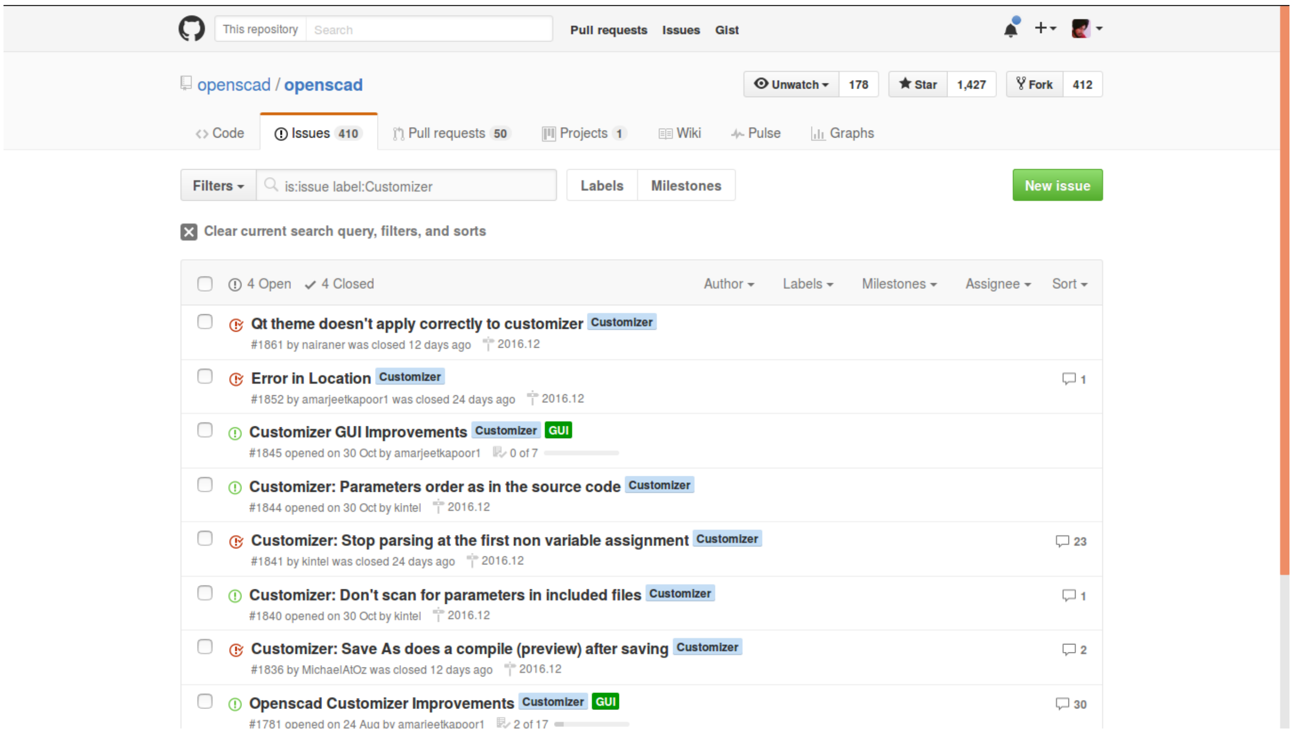
Elapsed time 22 min

about a month ago

Job log | View config

```
{
  "language": "cpp",
  "before_install": [
    "if [[ \"$DIST\" == \"trusty\" ]]; then wget -qO - http://files.openscad.org/OBS-Repository-Key.pub | sudo apt-key add - ; echo 'yes' | sudo a
    "if [[ \"$DIST\" == \"precise\" ]]; then echo 'yes' | sudo add-apt-repository ppa:chrysn/openscad ; sudo apt-get update -qq ; sudo apt-get pur
    "if [[ \"$TRAVIS_OS_NAME\" == \"osx\" ]]; then brew update ; ./scripts/macosx-build-homebrew.sh ; fi"
  ],
  "before_script": [
    "echo -e \"Host files.openscad.org\\n\\tStrictHostKeyChecking no\\n\" >> ~/.ssh/config",
    "for i in {0..40}; do eval $(printf \"echo \\$openscad_rsa_%02d\\n\" $i) >> ~/.ssh/openscad_rsa_base64; done",
    "base64 --decode ~/.ssh/openscad_rsa_base64 > ~/.ssh/id_rsa",
    "chmod 600 ~/.ssh/id_rsa"
  ],
  "script": "if [ \"${COVERITY_SCAN_BRANCH}\" != 1 ]; then ./scripts/travis-ci.sh ; fi",
  "global_env": "OPENSCAD_UPLOAD_TESTS=yes openscad_rsa_00=[secure] openscad_rsa_01=[secure] openscad_rsa_02=[secure] openscad_rsa_03=[secure] ope
  "group": "stable",
  "dist": "precise",
  "cache": "apt",
  "compiler": "gcc",
  "env": "DIST=\"precise\"",
  "os": "linux"
```

Figure 4.24: Configuration and summary of single enviorment



This repository Search | Pull requests Issues Gist

openscad / openscad — Unwatch 178 | Star 1,427 | Fork 412

Code | Issues 410 | Pull requests 50 | Projects 1 | Wiki | Pulse | Graphs

Filters | is:issue label:Customizer | Labels | Milestones | New issue

Clear current search query, filters, and sorts

4 Open ✓ 4 Closed — Author, Labels, Milestones, Assignee, Sort

- Qt theme doesn't apply correctly to customizer `Customizer` — #1861 by nairaner was closed 12 days ago 2016.12
- Error in Location `Customizer` — #1852 by amarjeetkapoor1 was closed 24 days ago 2016.12 — 1
- Customizer GUI Improvements `Customizer` `GUI` — #1845 opened on 30 Oct by amarjeetkapoor1 0 of 7
- Customizer: Parameters order as in the source code `Customizer` — #1844 opened on 30 Oct by kintel 2016.12
- Customizer: Stop parsing at the first non variable assignment `Customizer` — #1841 by kintel was closed 24 days ago 2016.12 — 23
- Customizer: Don't scan for parameters in included files `Customizer` — #1840 opened on 30 Oct by kintel 2016.12 — 1
- Customizer: Save As does a compile (preview) after saving `Customizer` — #1836 by MichaelAtOz was closed 12 days ago 2016.12 — 2
- Openscad Customizer Improvements `Customizer` `GUI` — #1781 opened on 24 Aug by amarjeetkapoor1 2 of 17 — 30

Figure 4.25: List of Issues provide by Alpha and Beta testers

Table 4.1: Table to show various Tests

Test cases for customizer				
Sr.	Test Name	Test Description	No. of test Cases	Status
1	customizertest_description	Check if description syntax work	24	Pass
2	customizertest_parameter	Check if parameter syntax work	33	Pass
3	customizertest_allmodulescomment	Check iteraction of new syntax with modules	39	Pass
4	customizertest_allfunctionscomment	Check iteraction of new syntax with functionas	33	Pass
5	customizertest_allexpressionscomment	Check iteraction of new syntax with expressions	37	Pass
6	customizertest_group	Check if group syntax work	16	Pass
7	customizertest-first_setofparameter	Check if Json is read correctly	1	Pass
8	customizertest-wrong_setofparameter	Check if Json is read correctly for wrong syntax	1	Pass
9	customizertest-incomplete_setofparameter	Check if Json is read correctly for incomplete parameters	1	Pass
10	customizertest-imgset_setofparameter	Check if Json is read correctly for imaginary set	1	Pass

CHAPTER 5 CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

This project was something which required in-depth view in designing the programming language and analysis of your problems. The main problem in this project was encountered during this project was related to writing a parser that is able to parse the context sensitive grammar while also keeping the big chunk of context-free grammar part of language separate. Even with this major problem we were able to finish this project before the set time and provide the user with the work which they can use, test and provide feedback.

A great deal of things are learned while working on this project. The learning was not limited to project only but the whole experience of working as a trainee under Dr. H.S. Rai at TCC, developer at OpenSCAD and also as a mentor for Google Code In was immensely educational. Working with the Open Source Community and a variety of people of different age group, one is always challenged by the fundamental difference between classroom coaching and real World experience. But such a challenge is exactly the purpose of six months training.

The whole experience of working on this project and contributing to a few others has been very rewarding as it has given great opportunities to learn new things and get a firmer grasp on already known technologies. Here is a reiteration of some of the technologies I have encountered, browsed and learned:

1. Operating System: Linux
2. Language: C++, flex, bison, JSON, LaTeX
3. FrameWork: Qt, Travis CI
4. Softwares: Git, cTest, GRASS GIS, Doxygen
5. Building Tools: qmake, CMake, make
6. Communication tools: IRC

So during this project, I learned all the above things. Above all, I got to know how software is developed and how much work and attention to details is required in building even the most basic of components of any project. Apart from above I also learned things like:

1. Planning
2. Designing
3. Developing code
4. Working in a team
5. Testing
6. Licensing Constraints
7. How Open source community work?

8. Writing Readable code
9. Coding Standards

And these are all very precious lessons in themselves.

5.2 Future Scope

OpenSCAD being an open source project and supported by a large open Source community have a lot of scope for future improvements and additions as other individuals can also contribute in it and add additional functionality. One of the examples is my project only customizer for OpenSCAD.

Being an Open Source project there is a constant flow of suggestion and demands by people for additional functionality and improvement in existing features. Customizer being no exception constant flow of feature of suggestions and feature requests even before the start of development. Here is a small list of the Features that would be added in near future.

1. **Adding support for InBuid syntax:** It was decided much at beginning of the project that this feature will be ported using a special syntax then relying on the comment based syntax. The discussion has been going on related to deciding the syntax for the customizer.
2. **Option to Add images:** It has been proposed that there should be a feature to add images through customizer.
3. **Option to Include files with Customizer:** This feature will help people change the include file with the help of Customizer which could provide a feature which would be related to Run Time Polymorphism.
4. **Conditional display of Parameter in customizer:** Sometimes people want to show a different set of parameters based on the chooses made by the user before that parameter or sometimes we just want to hide the parameters because of the selection made by the user in the pervious parameter.
5. **Improving GUI part:** There have been some suggestions related to improving the GUI of the Customizer https://github.com/openscad/openscad/issues/1845 .
6. **Providing syntax to manipulate modules parameter:** This feature will help user pick a customized version of the module using the customizer. They will be able to Customize the parameters of modules using and the Customizer than drag that into the file to call it with customized parameter.

Apart from these features, their are Some suggestion related to how existing feature would be improved to refer to that you can visit following:

1. https://github.com/openscad/openscad/issues/1844
2. https://github.com/openscad/openscad/issues/1840
3. https://github.com/openscad/openscad/issues/1781
4. https://github.com/openscad/openscad/issues/722

BIBLIOGRAPHY

[1] J. Levine, Flex & bison, 1st ed. Sebastopol, Calif.: O'Reilly Media, 2009.

[2] C. Donnelly and R. Stallman, Bison, 1st ed. Boston, MA: Free Software Foundation, 1999.

[3] "User:Amarjeet Singh Kapoor/GSoC2016/Project - BRL-CAD", Brlcad.org, 2016. [Online]. Available: http://brlcad.org/wiki/User:Amarjeet_Singh_Kapoor/GSoC2016/Project. [Accessed: 27- Nov- 2016].

[4] "User Interface for Customizing Models (Part 3)", amarjeetkapoor1, 2016. [Online]. Available: https://amarjeetkapoor1.wordpress.com/2016/08/17/user-interface-for-customizing-models-part-3/. [Accessed: 27- Nov- 2016].

[5] "OpenSCAD User Manual/WIP - Wikibooks, open books for an open world", En.wikibooks.org, 2016. [Online]. Available: https://en.wikibooks.org/wiki/OpenSCAD_User_Manual/WIP#Customizer. [Accessed: 27- Nov- 2016].

[6] "openscad/openscad", GitHub, 2016. [Online]. Available: https://github.com/openscad/openscad/wiki/Project%3A-Form-based-script-parameterization. [Accessed: 27- Nov- 2016].

[7] "User Interface for Customizing Models (Part 2)", amarjeetkapoor1, 2016. [Online]. Available: https://amarjeetkapoor1.wordpress.com/2016/07/18/user-interface-for-customizing-models-part-2/. [Accessed: 27- Nov- 2016].

[8] "openscad/openscad", GitHub, 2016. [Online]. Available: https://github.com/openscad/openscad. [Accessed: 27- Nov- 2016].

[9] "Doxygen: Main Page", Doxygen.org, 2016. [Online]. Available: http://www.doxygen.org. [Accessed: 27- Nov- 2016].

[10] "OpenSCAD", En.wikipedia.org, 2016. [Online]. Available: https://en.wikipedia.org/wiki/OpenSCAD. [Accessed: 27- Nov- 2016].

[11] http://www.openscad.org/

[12] "MakerBot Customizer", Customizer.makerbot.com, 2016. [Online]. Available: http://customizer.makerbot.com/docs. [Accessed: 27- Nov- 2016].

[13] "User Interface for Customizing Models", amarjeetkapoor1, 2016. [Online]. Available: https://amarjeetkapoor1.wordpress.com/2016/07/04/user-interface-for-customizing-models/. [Accessed: 27- Nov- 2016].

[14] "OpenSCAD - News", Openscad.org, 2016. [Online]. Available: http://www.openscad.org/news.html#20160714. [Accessed: 27- Nov- 2016].

[15] "GNU", En.wikipedia.org, 2016. [Online]. Available:

https://en.wikipedia.org/wiki/GNU#/media/File:Heckert_GNU_white.svg. [Accessed: 27- Nov- 2016].

[16] "Qt (software)", En.wikipedia.org, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Qt_(software)#/media/File:Qt_logo_2015.svg. [Accessed: 27- Nov- 2016].

[17] "",Upload.wikimedia.org, 2016. [Online]. Available:

https://upload.wikimedia.org/wikipedia/commons/c/ce/Doxygen.png. [Accessed: 27- Nov- 2016].

[18] "Git", En.wikipedia.org, 2016. [Online]. Available:

https://en.wikipedia.org/wiki/Git#/media/File:Git-logo.svg. [Accessed: 27- Nov- 2016].

[19] "Qt (software)", En.wikipedia.org, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Qt_(software). [Accessed: 27- Nov- 2016].

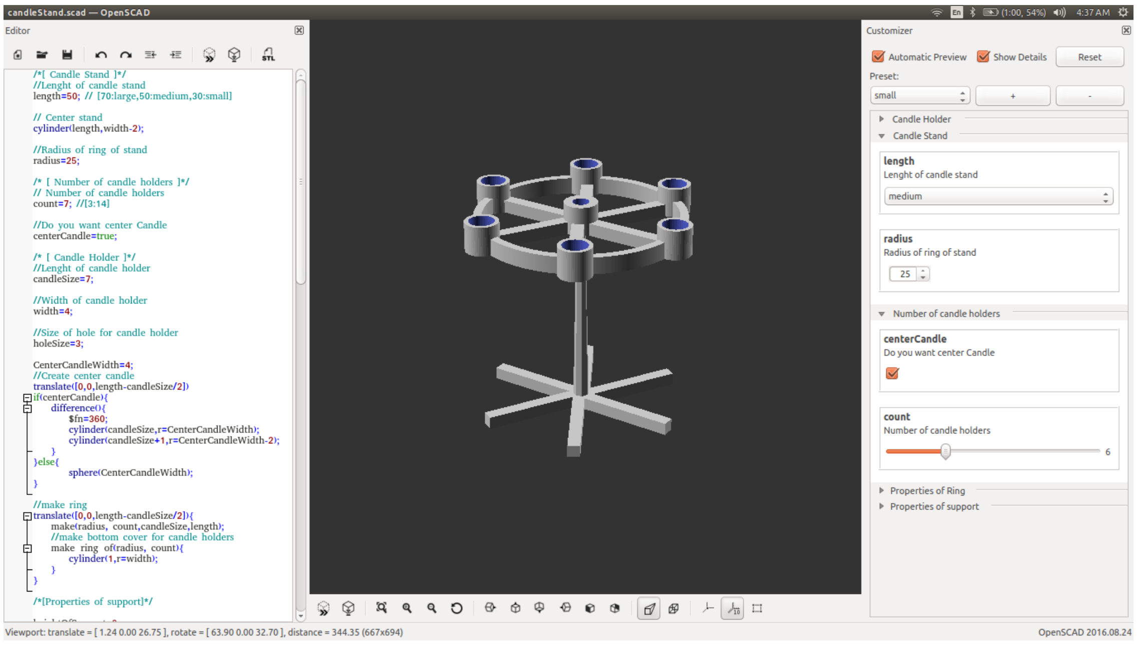
[20] "C++", En.wikipedia.org, 2016. [Online]. Available: https://en.wikipedia.org/wiki/C%2B%2B. [Accessed: 27- Nov- 2016].

[21] "Travis CI", En.wikipedia.org, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Travis_CI. [Accessed: 27- Nov- 2016].

[22] "CMake/Testing With CTest - KitwarePublic", Cmake.org, 2016. [Online]. Available: https://cmake.org/Wiki/CMake/Testing_With_CTest. [Accessed: 27- Nov- 2016].

APPENDIX A IMPLEMENTATION EXAMPLE

Example 1: To Make Parameteric candle Stand model in OpenSCAD using customizer



```
/*[ Candle Stand ]*/
//Lenght of candle stand
length=50; // [70:large,50:medium,30:small]

// Center stand
cylinder(length,width-2);

//Radius of ring of stand
radius=25;

/* [ Number of candle holders ]*/
// Number of candle holders
count=7; //[3:14]

//Do you want center Candle
centerCandle=true;

/* [ Candle Holder ]*/
//Lenght of candle holder
candleSize=7;

//Width of candle holder
width=4;

//Size of hole for candle holder
holeSize=3;

CenterCandleWidth=4;
//Create center candle
translate([0,0,length-candleSize/2])
if(centerCandle){
    difference(){
        $fn=360;
        cylinder(candleSize,r=CenterCandleWidth);
        cylinder(candleSize+1,r=CenterCandleWidth-2);
    }
}else{
        sphere(CenterCandleWidth);
}

//make ring
translate([0,0,length-candleSize/2]){
    make(radius, count,candleSize,length);
    //make bottom cover for candle holders
    make ring of(radius, count){
        cylinder(1,r=width);
    }
}

/*[Properties of support]*/
```

Candle Stand:
- length — Lenght of candle stand: medium
- radius — Radius of ring of stand: 25

Number of candle holders:
- centerCandle — Do you want center Candle: ✓
- count — Number of candle holders: 6

Viewport: translate = [1.24 0.00 26.75], rotate = [63.90 0.00 32.70], distance = 344.35 (667x694)

Figure A.1: Output of Code for Candle Stand model

```
/*[ Candle Stand ]*/
//Lenght of candle stand
length=50; // [70:large,50:medium,30:small]
// Center stand
cylinder(length,width-2);
//Radius of ring of stand
radius=25;

/* [ Number of candle holders ]*/
// Number of candle holders
count=7; //[3:14]
//Do you want center Candle
centerCandle=true;

/* [ Candle Holder ]*/
//Lenght of candle holder
candleSize=7;
//Width of candle holder
width=4;
//Size of hole for candle holder
holeSize=3;
CenterCandleWidth=4;
```

```
/*[Properties of support]*/

heightOfSupport=3;
widthOfSupport=3;

/*[Properties of Ring]*/

heightOfRing=4;
widthOfRing=23;


//Create center candle
translate([0,0,length-candleSize/2])
if(centerCandle){
 difference(){
     $fn=360;
     cylinder(candleSize,r=CenterCandleWidth);
     cylinder(candleSize+1,r=CenterCandleWidth-2);
 }
}else{
     sphere(CenterCandleWidth);
}

//make ring
translate([0,0,length-candleSize/2]){
 make(radius, count,candleSize,length);
 //make bottom cover for candle holders
 make_ring_of(radius, count){
     cylinder(1,r=width);
 }
}

//Base of candle stand
for (a = [0 : count - 1]) {
 rotate(a*360/count) {
 translate([0, -width/2, 0])
     cube([radius, widthOfSupport, heightOfSupport]);
 }
}

//make ring with candle holders
module make(radius, count,candleSize,length){

 $fa = 0.5;
 $fs = 0.5;
 difference(){
     union(){
            //making holders
            make_ring_of(radius, count){
                cylinder(candleSize,r=width);
            }

            //Attaching holders to stand
            for (a = [0 : count - 1]) {
                rotate(a*360/count) {
                translate([0, -width/2, 0])
                    cube([radius, widthOfSupport, heightOfSupport]);
                }
            }

            //make ring
            linear_extrude(heightOfRing)
            difference(){
                circle(radius);
                circle(widthOfRing);
            }
        }
        //Making holes in candle holder
        make_ring_of(radius, count){
            cylinder(candleSize+1,r=holeSize);
        }
    }
}


module make_ring_of(radius, count){
 for (a = [0 : count - 1]) {
     angle = a * 360 / count;
     translate(radius * [cos(angle), -sin(angle), 0])
             children();
 }
}

// Written by Amarjeet Singh Kapoor <amarjeet.kapoor1@gmail.com>
//
// To the extent possible under law, the author(s) have dedicated all
// copyright and related and neighboring rights to this software to the
// public domain worldwide. This software is distributed without any
// warranty.
//
// You should have received a copy of the CC0 Public Domain
// Dedication along with this software.
// If not, see <http://creativecommons.org/publicdomain/zero/1.0/>.
```