MIDDLE EAST TECHNICAL UNIVERSITY
COMPUTER ENGINEERING
DEPARTMENT

SOFTWARE DESIGN DESCRIPTION

Group Name : Smeshers
Group Members : Uğur Yanıkoğlu
Furkan Odluyurt
Dicle Ayzit
Emre Barış

Advisors : Yusuf Sahillioğlu
Çağlar Seylan

Design Description of Digital Geometry Processing Toolkit, Meshtika.

MESHTIKA

1.INTRODUCTION
- 1.1 Purpose
- 1.2 Scope
- 1.3 Intended Audience
- 1.4 Definitions, Acronyms & Abbreivations
- 1.5 Date of Issue and Status
- 1.6 Issuing Organization
- 1.7 Authorship
- 1.8 Design Languages
- 1.9 Context
- 1.10 Summary

2. REFERENCES
3. DESIGN CONCERNS
4. DESIGN VIEWPOINT
- 4.1. Introduction
- 4.2 Context Viewpoint
  - 4.2.1 Use Case Diagram and Use Case Definitions
- 4.3 Composition Viewpoint
  - 4.3.1. Functional (Logical) Decomposition Viewpoint
  - 4.3.2. Run-Time (Physical) Decomposition Viewpoint
- 4.4 Logical Viewpoint
- 4.5 Dependency Viewpoint
- 4.6 Information Viewpoint
- 4.7 Patterns Use Viewpoint
- 4.8 Interface Viewpoint
- 4.9 Structure Viewpoint
- 4.10 Interaction Viewpoint

5.DESIGN RATIONALE

Table of Figures

Figure 1 :…………………………………………………….……………………………….. 7
Figure 2 : ………………………………….……...9
Figure 3 : …………………..……………………………………………………....11
Figure 4 : ……………..………………………..…………………………………......13
Figure 5 : ……………………………………...……………………………….....14
Figure 6 : …..……………..……………..……………….………………………….....15

List of Tables

Table 1 :…………………………………………………….……………………………….. 4
Table 2 :…………………………………………………….……………………………….. 8

1.INTRODUCTION

1.1 Purpose

The purpose of this document is to describe and visualize the architecture and system design of Meshtika by using different viewpoints.

The purpose of this document is to describe the software system which is planned to meet the needs specified in Software Requirements Specification document. This document will be reference to the coding phase.

1.2 Scope

This document includes a structural overview of all data, modules and interfaces. Besides that , it covers design of each modules in detail through giving information about the comprehensive software architecture. A bunch of design views will be presented in order to support the design and development process. This documentation will be a guide during the entire implementation phase.

1.3 Intended Audience

The intended audience for this document is the group members of the Smeshers and stakeholders of the project.

The stakeholders of this project are :

- Team Leader of the project - Çağlar Seylan
- Supervisor of the project - Yusuf Sahillioğlu
- The Project Managers - Attila Özgit , Emre Akbaş
- The Evaluation Committee of the CENG DEMO Day

1.4 Definitions, Acronyms & Abbreivations

DGP	Digital Geometry Processing
GUI	Graphical User Interface.
API	Application Programming Interface
IEEE	Institute of Electrical and Electronics Engineers.
IDE	Integrated Development Environment

SRS	Software Requirements Specification
SDD	Software Design Description
STD	Software Test Document
WYSIWYI	What You See is What You Implemented
SSD	Step by Step Debugger (Software Component)
Mesh	A data structure in Computer Graphics
Output	Anything written to Info is output

Table 1: Table of Abbreviations

1.5 Date of Issue and Status

The date of issue of the initial version of this document is 03.06.2016. Changes may be done with respect of the stakeholders’ or open source communitites’ will in the future.

1.6 Issuing Organization

The issuing organization of this project is Middle East Technical University Computer Engineering Department. The applicability of this project will be evaluated by them.

1.7 Authorship

All sections in this document is written by group members of Smeshers. The authors of this document are responsible for updating it.

1.8 Design Languages

Unified Modeling Language (UML) is used to represent the design properties of the Meshtika System.

1.9 Context

This document contains the software design descriptions for BISIM system. This document is prepared according to “IEEE Standart for Information Technology - Systems Design - IEEE 1016 2009”.

This document releases the details of how Meshtika system is to be built. The details are represented through graphical notations such as use case models, class diagrams, viewpoints and other supporting design information.

1.10 Summary

This documentation is intended for anyone who wishes to understand how software Meshtika was designed internally. All system constraints, functionalities and elements included in the system are explained in detail. By using this documentation, developers can collaborate in order to shape the project to the final specification or by using this document, they can extend the boundaries of the system for further demand.

The intoduction about this documentation and the system have been made in previous sections. Mentioned representations of the system's design , design concerns and design rationale will be explained in next sectios respectively.

2. REFERENCES

- IEEE 10162009, IEEE Standard for InformationTechnology-Systems Design-Software Design Descriptions

- Software Requirements Specification Document for Meshtika (2015), from http://senior.ceng.metu.edu.tr/2016/smeshers/

- Software Design Description Document for Meshtika (2016), from http://senior.ceng.metu.edu.tr/2016/smeshers/

- Blender Manual Contents (n.d.). Retrieved January 13, 2016, from https://www.blender.org/manual/

- Blender.org - Home of the Blender project - Free and Open 3D Creation Software. (n.d.). Retrieved January 13, 2016, from https://www.blender.org/

3. DESIGN CONCERNS

There are several design concerns of the system. They will be mentioned respectively with their possible solutions in this section.

One of these concerns is the problem related to integration of the Meshtika API. As it is known, Meshtika is based on Blender. It is possible to make addons for Blender. However , integrating a module to existing application is quite a problem. This problem can only be solved by changing the initialization scripts of the existing program and recompling whole system.

Another concern is the performance of the system and algorithms; they should give real time responses to the users. Time intervals between actions and their responses must not pass the limitations. The bugs inside a huge application that affect the performance is a big problem for only one team containing 4 members but these bugs will be tried to fixed as far as possible. The DGP algorithms will be implemented directly from their respective papers and they will be optimized as far as possible.

Yet another concern is adding and removing GUI components to develop easy to use application. It is expected to solve this issue only modifying python side of the Blender. However, some components and functionalities is bounded in the C/C++ side of the application. To add new functionalities , similarly , the new functionality have to be bounded to related functionality in C/C++ side. As it is seen , the developers of this project have to examine the C/C++ side of the application as well since this is only solution for that particular problem.

4. DESIGN VIEWPOINT

4.1. Introduction

In this section of the sdd, following nine viewpoints will be provided with detailed description and diagrams.

- Context Viewpoint
- Composition Viewpoint
- Logical viewpoint
- Dependency viewpoint

- Information viewpoint
- Patterns use viewpoint
- Interface viewpoint
- Structure viewpoint
- Interaction viewpoint

4.2 Context Viewpoint

Context viewpoint consists of the relationships, interactions and dependencies between the enviorement and the system. Use case diagram and use case descriptions are presented to give a general sense of the context viewpoint.

4.2.1 Use Case Diagram and Use Case Definitions

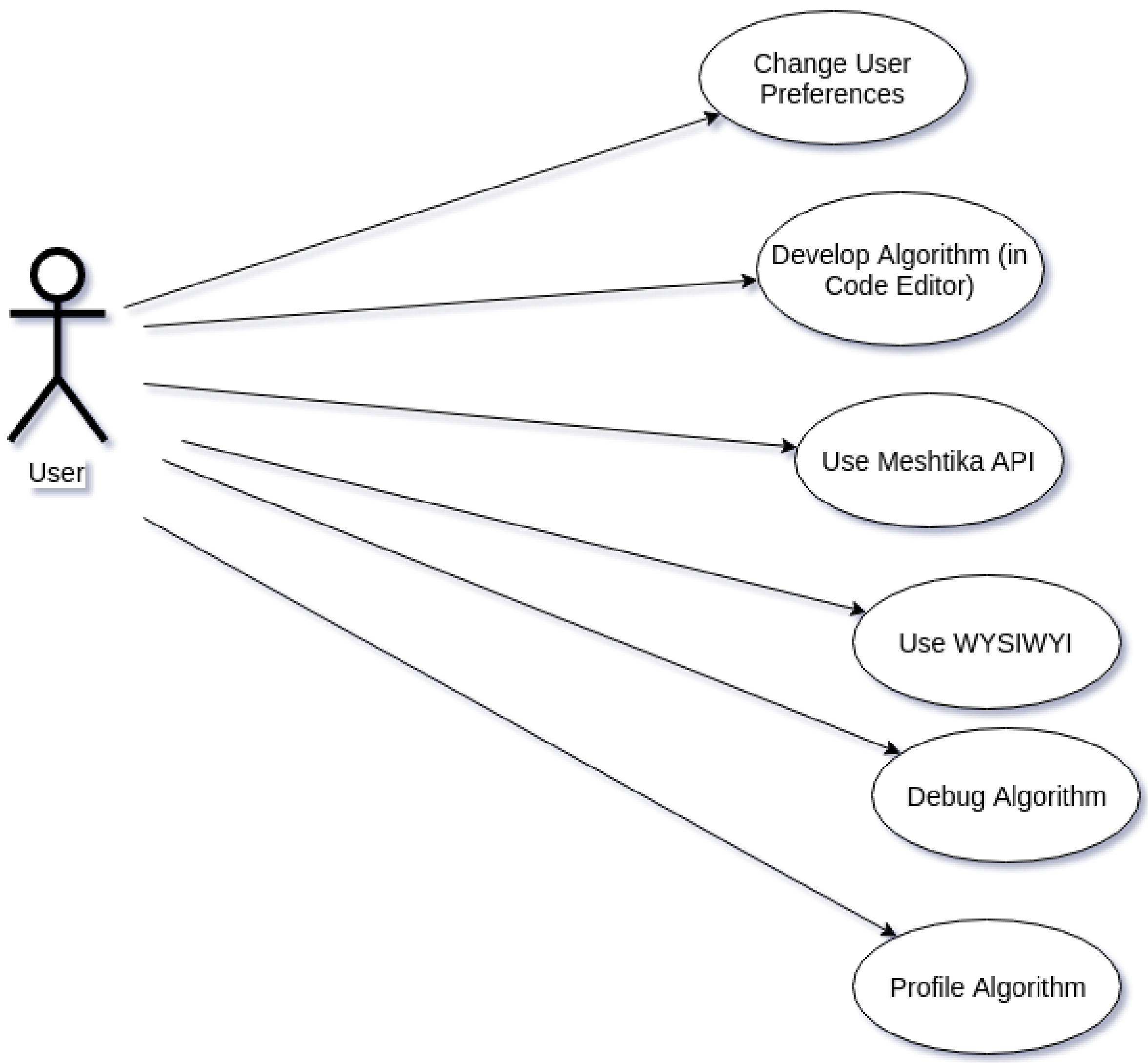


Figure 1 : Use Case Diagram of Meshtika

Use-case Name	**Definition**
Develop Algorithm	Develop a custom DGP algorithm in the environment. IDE offers a robust text editor for this purpose
Debug Algorithm	Detect the logical errors the custom algorithm may have by classic step-by-step debugging technique
Use Meshtika API	Develop a custom DGP algorithm in the environment by an offered, easy-to-use API
Change User Preferences	Users of Meshtika will have the power of customizing the platform, respecting to certain limits, via user preferences. They can perform actions, such as personalizing the theme of text editor, installing/ enabling/disabling add-ons, which are the correspondents of classic plug-ins in the scope of Blender
Use WYSIWYI	Analyzing algorithm by changing its parameters
Profile Algorithm	Observing time and space complexities of algorithms

Table 2 : Use Case Definitions of Meshtika

The elaborated descriptions, including actors, triggers, alternative scenarios, pre and post conditions for each use case was already given in the Software Requirement Specification Document.

4.3 Composition Viewpoint

In this section, project components and connections between these components are explained. To understand the composition viewpoint of the system, component and deployment diagrams are provided.

4.3.1. Functional (Logical) Decomposition Viewpoint

The first viewpoint is the functional (logical) decomposition viewpoint. To represent the outcome of applying the logical viewpoint, UML component diagram can be made use of, since component diagrams let the designers model the internal structure of the system regarding the major components.

As displayed in the component diagram below, a component is what is required to execute a stereotype function, such as executables, documents, database tales, library files, etc. Moreover, the components require interfaces to be able to interact with one another. For Meshtika, the major internals are Code Editor, GUI, Meshtika API, WYSIWYI, Profiler and Debug components, since all these are necessary to keep the system active and responding. More detailed explanation of the interfaces will be given in the Interface Viewpoint section, referring to this component diagram.

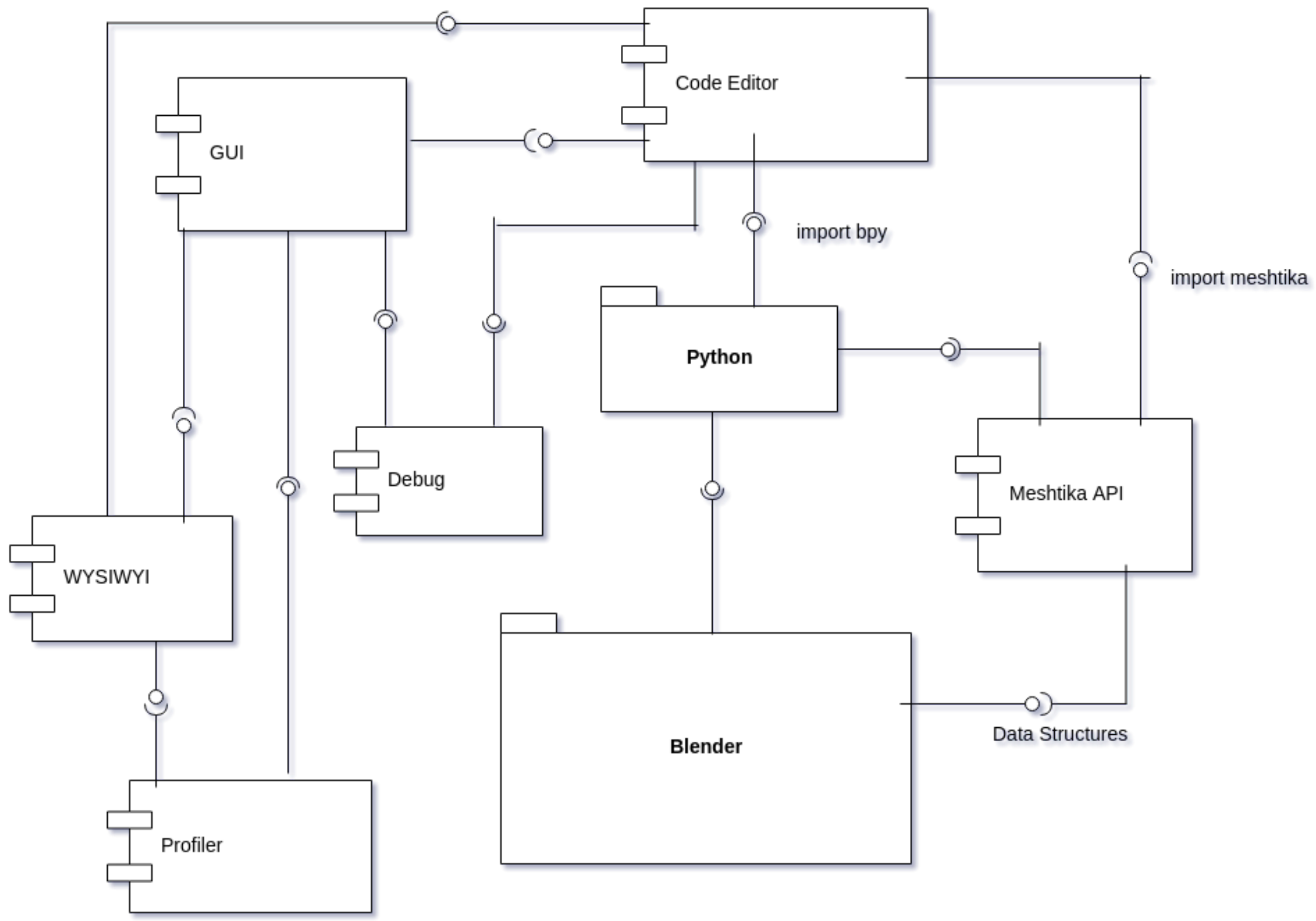


Figure 2: Component Diagram of Meshtika

To make things clear, here are the brief explanations of the major components shown in the diagram above.

GUI is the user interface of the program. It actually includes the Code Editor, the Info Space, the 3D View and the WYSIWYI environment.

Code Editor is the text editor, where the developers are expected to write their scripts and manipulate them via the features provided by the Code Editor.

Info Space is the part where the log of the actions taken in Code Editor and non-visual result of run script is displayed.

3D View is the part to visualize the triangular meshes.

WYSIWYI is the part where a list of registered parameters in the script in the code editor is displayed and it is possible to alter these parameters, without altering the script itself and explore their outcomes, such as the change in resulting mesh or the numeric value output in the log.

Meshtika API is the package provided to the users, which is intended to make DGP development more comfortable, as several famous and state-of-the art papers' algorithms implementations are integrated. This can be easily imported in the script in the code editor, and be made use of.

Debugger is a classic debugger, letting the developer debug his/her code written in the code editor with classic debug functions provided.

Profiler is the component that provides the relation between the used recources and time as a graphical outcome.

4.3.2. Run-Time (Physical) Decomposition Viewpoint

The second viewpoint that is applied in the scope of composition viewpoint is the run-time (physical) decomposition viewpoint, which captures the physical (often hardware) components in the system and how these components are interconnected. There is also the deployment viewpoint, which captures how logical components are mapped onto physical ones. By the help of UML deployment diagram, it is possible to combine the deployment and physical decomposition viewpoints, since deployment diagrams display the execution architecture of systems, representing the deployment of software artifacts, where artifacts refer to concrete elements in the physical world, as an outcome of development and deployment. Thus, the deployment diagram of Meshtika and a detailed description of it is given below.

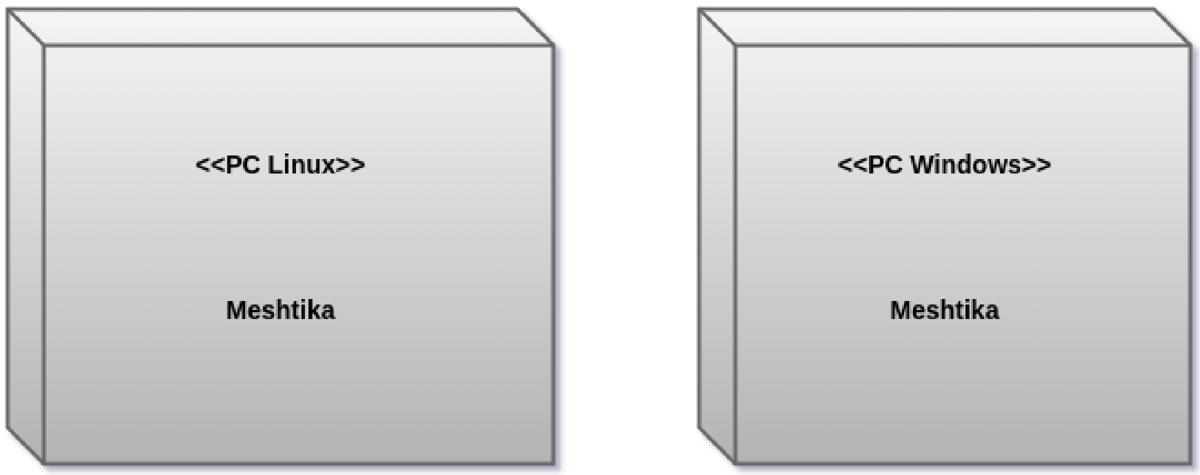


Figure 3: Deployment Diagram of Meshtika

As Meshtika is a stand-alone project, that does not require any interaction or dependency with the outside it can be adopted in PCs that is using either one of the operating systems: Linux and Windows.

4.4 Logical Viewpoint

The logical viewpoint describes the logical structure and the distribution of responsibilities functionality of a system by means of a network of interacting logical components that are responsible for a set of functions.

The class diagram is displayed in Software Requirement Specification Document of Meshtika to serve the insight of this viewpoint.

4.5 Dependency Viewpoint

The dependency viewpoint specifies the relationships of interconnection and access among entities. Execution ordering and data flow between shared information are included in this viewpoint. The component diagram given in the section 4.3.1 is an instance of this kind of viewpoint. It shows the relationships and provides an overall picture of subjects in order to help maintainers resolve the issues with the system. This can be done by examining every component and the relations between them.

To elaborate thses dependecies, here are given the explanations regarding the components and their relations.

First of all, Code Editor is actually a part of GUI of the program, therefore it is represented as Code Editor providing an interface to the GUI in the component diagram. If we get mor einto detail, it should be stated that the results of the actions performed in the Code Editor, such as running the script, will affect other parts of the GUI. For example, errors will pop up in the Info Space of the GUI and resulting mesh (if there will be any) will appear in the 3D view part of GUI.

Python provides the Blender/Python API (bpy) to Blender, which we also made use of extensively during our development. It makes it possible to access Blender's Python side source code and let it be manipulated by the developers, which is how we mostly altered the GUI of Blender.

Both bpy and meshtika API, which is the API provided for Meshtika users including several paper implementations to make DGP development easier for them, can be imported inside the script in the Code Editor by the developer. It should be noted that Meshtika API makes use of the data structures of Blender, such as mesh.

WYSIWYI represents the registered parameters, which is provided by the Code Editor. The alteration in WYSIWYI can be tracked in GUI, either info space or 3D view.

Debugger receives the script in code editor and traverses it. It outputs the log in Info Space of GUI.

Profiler also receives the script in the code editor and represents a graphical output in GUI.

4.6 Information Viewpoint

The ultimate purpose of any information system being to manipulate data, the information viewpoint describes the way an architecture stores, manipulates, manages and distributes information. Meshtika is not a system that stores any user related data or manipulation of it, though it responds to session related data as any IDE would, such as keeping the current script in its data system and make use of it while running script, making use of profiler and debugger. Moreover, with its integrated WYSIWYI environment, it again accesses the corresponding sessions' script's vital arguments to be manipulated over.

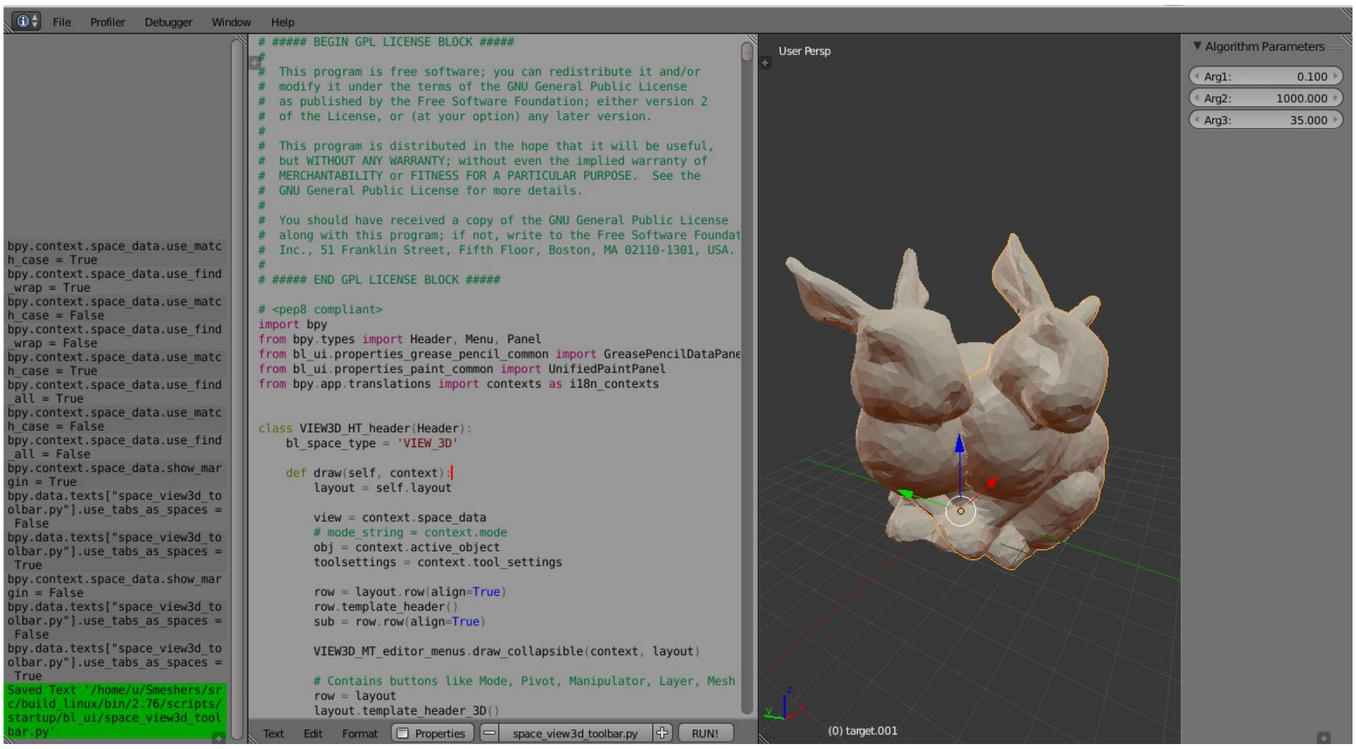
4.7 Patterns Use Viewpoint

In this viewpoint, design patterns are discussed. Meshtika embraces Model-View-Controller Pattern.

This model divides the system into three interconnected parts, so that information exchanged between the components of the system can get understandable by the user. The GUI part of the system is the view component of this pattern. Data structures provided by Blender are the model part of the system. The Meshtika API including the user-friendly implementations of famous algorithms served to the user is the controller part of the system.

4.8 Interface Viewpoint

In this section, the main GUI of Meshtika is provided, which includes from left to right, Info Space, Code Editor, 3D View and WYSIWYI parts, respectively.



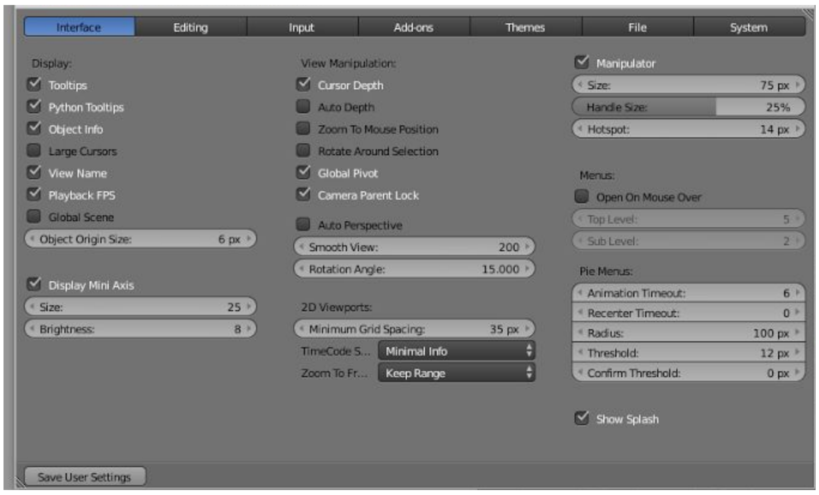
```
bpy.context.space_data.use_match_case = True
bpy.context.space_data.use_find_wrap = True
bpy.context.space_data.use_match_case = False
bpy.context.space_data.use_find_wrap = False
bpy.context.space_data.use_match_case = True
bpy.context.space_data.use_find_all = True
bpy.context.space_data.use_match_case = False
bpy.context.space_data.use_find_all = False
bpy.context.space_data.show_margin = True
bpy.data.texts["space_view3d_toolbar.py"].use_tabs_as_spaces = False
bpy.data.texts["space_view3d_toolbar.py"].use_tabs_as_spaces = True
bpy.context.space_data.show_margin = False
bpy.data.texts["space_view3d_toolbar.py"].use_tabs_as_spaces = False
bpy.data.texts["space_view3d_toolbar.py"].use_tabs_as_spaces = True
Saved Text '/home/u/Smeshers/src/build_linux/bin/2.76/scripts/startup/bl_ui/space_view3d_toolbar.py'
```

```
# ##### BEGIN GPL LICENSE BLOCK #####
#
#  This program is free software; you can redistribute it and/or
#  modify it under the terms of the GNU General Public License
#  as published by the Free Software Foundation; either version 2
#  of the License, or (at your option) any later version.
#
#  This program is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License for more details.
#
#  You should have received a copy of the GNU General Public License
#  along with this program; if not, write to the Free Software Foundat
#  Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.
#
# ##### END GPL LICENSE BLOCK #####

# <pep8 compliant>
import bpy
from bpy.types import Header, Menu, Panel
from bl_ui.properties_grease_pencil_common import GreasePencilDataPane
from bl_ui.properties_paint_common import UnifiedPaintPanel
from bpy.app.translations import contexts as i18n_contexts


class VIEW3D_HT_header(Header):
    bl_space_type = 'VIEW_3D'

    def draw(self, context):
        layout = self.layout

        view = context.space_data
        # mode_string = context.mode
        obj = context.active_object
        toolsettings = context.tool_settings

        row = layout.row(align=True)
        row.template_header()
        sub = row.row(align=True)

        VIEW3D_MT_editor_menus.draw_collapsible(context, layout)

        # Contains buttons like Mode, Pivot, Manipulator, Layer, Mesh
        row = layout
        layout.template_header_3D()
```

```
▼ Algorithm Parameters
Arg1:    0.100
Arg2: 1000.000
Arg3:   35.000
```

Figure 4: main GUI of Meshtika

There is just one other user interface that the user might deal with which is the user preferences screen, taht gives the user the freedom to select the features he/she wants in the scope of file type, interface an so on. User Preference screen is shown in the figure below.



Display:
- ✓ Tooltips
- ✓ Python Tooltips
- ✓ Object Info
- ▢ Large Cursors
- ✓ View Name
- ✓ Playback FPS
- ▢ Global Scene
- Object Origin Size: 6 px
- ✓ Display Mini Axis
- Size: 25
- Brightness: 8

View Manipulation:
- ✓ Cursor Depth
- ▢ Auto Depth
- ▢ Zoom To Mouse Position
- ▢ Rotate Around Selection
- ✓ Global Pivot
- ✓ Camera Parent Lock
- ▢ Auto Perspective
- Smooth View: 200
- Rotation Angle: 15.000

2D Viewports:
- Minimum Grid Spacing: 35 px
- TimeCode S... Minimal Info
- Zoom To Fr... Keep Range

✓ Manipulator
- Size: 75 px
- Handle Size: 25%
- Hotspot: 14 px

Menus:
- ▢ Open On Mouse Over
- Top Level: 5
- Sub Level: 2

Pie Menus:
- Animation Timeout: 6
- Recenter Timeout: 0
- Radius: 100 px
- Threshold: 12 px
- Confirm Threshold: 0 px

✓ Show Splash

Figure 5: User Preferences Screen of Meshtika

4.9 Structure Viewpoint

Making a detailed structure viewpoint here would be redundant since there many diagrams showing the structure of the system, such as in the section 4.3.1 component diagram and class diagram in SRS.

4.10 Interaction Viewpoint

This viewpoint releases the interaction information between the system entities and the actors. Interaction overview diagram is provided to understand this viewpoint clearly.

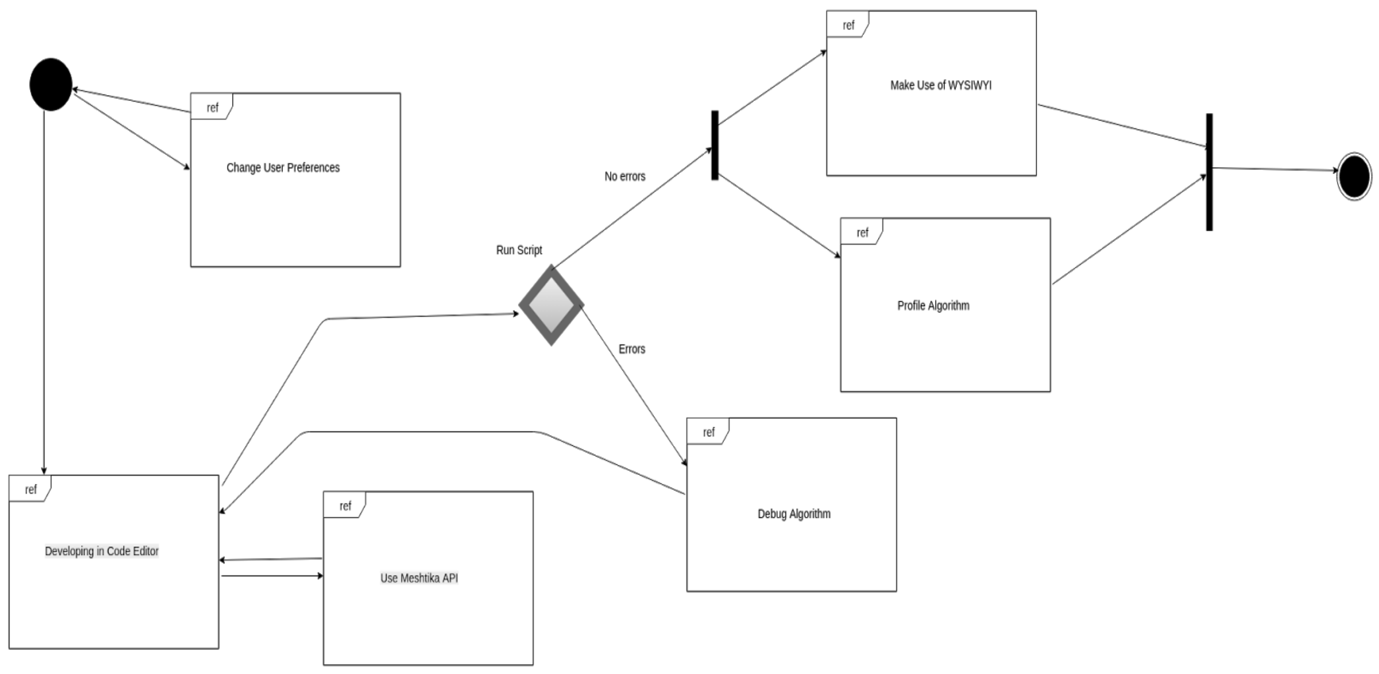


Figure 6: Interaction Overview Diagram of Meshtika

The ref frames in the interaction overview diagram are referenced to the use cases of the system, about which you can find the detailed information in SRS document of Meshtika.

It is important to note that Meshtika API can be used during the development phase. According to the script's being run without problems or not, the user is then able to debug the algorithm and get back to development again or can go on with the usage of WYSIWYI and profiling.

5.DESIGN RATIONALE

Design choices are created with respect to the specific requirements mentioned in SRS document and other significant features so that system can respond the required scenarios with specified cases. It can be updated according to changing requirements of the stakeholders and the users.

With the structure of this system, updates and upgrades will not cost much in cases of funds, efficiency and time.