Module Guide for Digital Twin Forest

Team 8, Forest Mirror
Tingyu Shi
Jiacheng Wu
Junhong Chen
Yichen Jiang
Bowen Zhang

April 2, 2023

1 Revision History

Date	Version	Notes
Jan 14	1.0	First Version
April 2	2.0	Final Draft

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

symbol	description
AC	Anticipated Change
DAG	Directed Acyclic Graph
M	Module
MG	Module Guide
OS	Operating System
R	Requirement
FR	Functional Requirement
NFR	Non-Functional Requirement
SC	Scientific Computing
SRS	Software Requirements Specification
Digital Twin Forest	Explanation of program name
UC	Unlikely Change
MVC	Model, Viewer, Controller
GUI	Graphical User Interface
LAI	Leaf Area Index
DBH	Diameter at breast height

Contents

List of Tables

List of Figures

3 Introduction

Decomposing a system into modules is a commonly accepted approach to developing software. A module is a work assignment for a programmer or programming team (Parnas et al., 1984). We advocate a decomposition based on the principle of information hiding (Parnas, 1972). This principle supports design for change, because the "secrets" that each module hides represent likely future changes. Design for change is valuable in SC, where modifications are frequent, especially during initial development as the solution space is explored.

Our design follows the rules layed out by Parnas et al. (1984), as follows:

- System details that are likely to change independently should be the secrets of separate modules.
- Each data structure is implemented in only one module.
- Any other program that requires information stored in a module's data structures must obtain it by calling access programs belonging to that module.

After completing the first stage of the design, the Software Requirements Specification (SRS), the Module Guide (MG) is developed (Parnas et al., 1984). The MG specifies the modular structure of the system and is intended to allow both designers and maintainers to easily identify the parts of the software. The potential readers of this document are as follows:

- New project members: This document can be a guide for a new project member to easily understand the overall structure and quickly find the relevant modules they are searching for.
- Maintainers: The hierarchical structure of the module guide improves the maintainers' understanding when they need to make changes to the system. It is important for a maintainer to update the relevant sections of the document after changes have been made.
- Designers: Once the module guide has been written, it can be used to check for consistency, feasibility, and flexibility. Designers can verify the system in various ways, such as consistency among modules, feasibility of the decomposition, and flexibility of the design.

The rest of the document is organized as follows. Section 4 lists the anticipated and unlikely changes of the software requirements. Section 5 summarizes the module decomposition that was constructed according to the likely changes. Section 6 specifies the connections between the software requirements and the modules. Section 7 gives a detailed description of the modules. Section 8 includes two traceability matrices. One checks the completeness of the design against the requirements provided in the SRS. The other shows the relation between anticipated changes and the modules. Section 9 describes the use relation between modules.

Notes: We use MVC as our software architecture, therefore, we decomposed our modules according to models, viewers, and controllers.

4 Anticipated and Unlikely Changes

This section lists possible changes to the system. According to the likeliness of the change, the possible changes are classified into two categories. Anticipated changes are listed in Section 4.1, and unlikely changes are listed in Section 4.2.

4.1 Anticipated Changes

Anticipated changes are the source of the information that is to be hidden inside the modules. Ideally, changing one of the anticipated changes will only require changing the one module that hides the associated decision. The approach adapted here is called design for change.

AC1: The UI appearance of the main page will be continuously upgraded.

AC2: The way of displaying environmental data and tree parameters may change.

AC3: The values of environmental data and tree parameters may change.

AC4: Tree parameters of different trees may change. For example, the stakeholders may add different tree parameters for different types of trees in the future.

AC5: The stakeholders may add more environmental data in the future.

4.2 Unlikely Changes

The module design should be as general as possible. However, a general system is more complex. Sometimes this complexity is not necessary. Fixing some design decisions at the system architecture stage can simplify the software design. If these decision should later need to be changed, then many parts of the design will potentially need to be modified. Hence, it is not intended that these decisions will be changed.

UC1: IO devices(keyboard & mouse).

UC2: The system will always run on macOS and Windows.

UC3: The 3D model of the virtual forest.

UC4: The way of viewing forest models.

5 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 1, 2, 3, and 4. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

Table 1: Module Hierarchy(Models)

Level 1	**Level 2**
Model Modules	M1: ForestTrees
	M2: ForestSky
	M3: ForestTerrain
	M4: RedPine
	M5: Oak
	M6: Beech
	M7: Birch
	M8: WhitePine
	M9: RedMaple
	M10: RedOak
	M11: EnvData
	M12: PlotData
	M13: FirstPersonPlayer
	M14: JsonFile

Table 2: Module Hierarchy(First Viewers Table)

Level 1	**Level 2**
Viewer Modules	M15: MainPageDisplay
	M16: StartButton
	M17: InstructionButton
	M18: ContactUsButton
	M19: QuitButton
	M20: InstructionInfoDisplay
	M21: ContactUsInfoDisplay
	M22: BackButton
	M23: UpdateDataDisplay
	M24: EnvDataSelectionButton
	M25: DataTypeSelectionButtons
	M26: NewDataInputBox
	M27: SaveButton

Table 3: Module Hierarchy(Second Viewers Table)

Level 1	**Level 2**
Viewer Modules	M28: CurrentDataDisplay
	M29: PlotSelectionDropDown
	M30: TreeTypeSelectionDropDown
	M31: UpdateDataButton
	M32: ForestDisplay
	M33: ShowEnvDataButton
	M34: ShowTreeParamButton
	M35: EnvDataDisplay
	M36: TreeParamDisplay
	M37: PauseIndicatorDisplay

Table 4: Module Hierarchy(Controllers)

Level 1	**Level 2**
Controller Modules	M38: JsonFileReader
	M39: JsonFileWriter
	M40: PauseManager
	M41: PlayerMovement
	M42: NewDataInputBoxController
	M43: StartButtonController
	M44: InstructionButtonController
	M45: UpdateDataButtonController
	M46: ContactUsButtonController
	M47: QuitButtonController
	M48: BackButtonController
	M49: PlotSelectionDropDownController
	M50: TreeTypeSelectionDropDownController
	M51: ShowEnvDataButtonController
	M52: ShowTreeParamButtonController
	M53: EnvDataSelectionButtonController
	M54: DataTypeSelectionButtonsController
	M55: SaveButtonController

6 Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements developed in the SRS(click here for revised SRS). In this stage, the system is decomposed into modules. The connection between functional requirements and modules is listed in Table 5. The connection between non-functional requirements and modules is listed in Table 6 and 7.

7 Module Decomposition

Modules are decomposed according to the principle of "information hiding" proposed by Parnas et al. (1984). The *Secrets* field in a module decomposition is a brief statement of the design decision hidden by the module. The *Services* field specifies *what* the module will do without documenting *how* to do it. For each module, a suggestion for the implementing software is given under the *Implemented By* title.

- If the entry is *OS*, this means that the module is provided by the operating system or by standard programming language libraries.
- If the entry is *Unity*, this means that the module is provide by libraries from Unity Engine.
- *Digital Twin Forest* means the module will be implemented by the Digital Twin Forest software

7.1 Model Modules

7.1.1 ForestTrees Module (M1)

Secrets: How the tree models are displayed.

Services: Provide virtual trees for the virtual forest.

Implemented By: Digital Twin Forest and Unity(Unity libraries are used to help build tree models)

7.1.2 ForestSky Module (M2)

Secrets: How the skybox and light are displayed.

Services: Provide virtual sky for the virtual forest.

Implemented By: Digital Twin Forest Unity(Unity libraries are used to help build forest sky)

7.1.3 ForestTerrain Module (M3)

Secrets: How the ground textures are displayed.

Services: Provide virtual terrain for the virtual forest.

Implemented By: Digital Twin Forest Unity(Unity libraries are used to help build forest terrain)

7.1.4 RedPine Module (M4)

Secrets: Attributes of red pine trees and associated methods about these attributes.

Services: Allow developers to define objects of red pine trees. From the objects, developers can store, read, or reset attributes about the red pine trees.

Implemented By: Digital Twin Forest

7.1.5 Oak Module (M5)

Secrets: Attributes of oak trees and associated methods about these attributes.

Services: Allow developers to define objects of oak trees. From the objects, developers can store, read, or reset attributes about the oak trees.

Implemented By: Digital Twin Forest

7.1.6 Beech Module (M6)

Secrets: Attributes of beech trees and associated methods about these attributes.

Services: Allow developers to define objects of beech trees. From the objects, developers can store, read, or reset attributes about the beech trees.

Implemented By: Digital Twin Forest

7.1.7 Birch Module (M7)

Secrets: Attributes of birch trees and associated methods about these attributes.

Services: Allow developers to define objects of birch trees. From the objects, developers can store, read, or reset attributes about the birch trees.

Implemented By: Digital Twin Forest

7.1.8 WhitePine Module (M8)

Secrets: Attributes of white pine trees and associated methods about these attributes.

Services: Allow developers to define objects of white pine trees. From the objects, developers can store, read, or reset attributes about the white pine trees.

Implemented By: Digital Twin Forest

7.1.9 RedMaple Module (M9)

Secrets: Attributes of red maple trees and associated methods about these attributes.

Services: Allow developers to define objects of red maple trees. From the objects, developers can store, read, or reset attributes about the red maple trees.

Implemented By: Digital Twin Forest

7.1.10 RedOak Module (M10)

Secrets: Attributes of red oak trees and associated methods about these attributes.

Services: Allow developers to define objects of red oak trees. From the objects, developers can store, read, or reset attributes about the red oak trees.

Implemented By: Digital Twin Forest

7.1.11 EnvData Module (M11)

Secrets: Attributes of plot environment and associated methods about these attributes.

Services: Allow developers to define objects of plot environment. From the objects, developers can store, read, or reset attributes about the plot environment.

Implemented By: Digital Twin Forest

7.1.12 PlotData Module (M12)

Secrets: Attributes of a plot and associated methods about these attributes. Attributes for a plot include plot environment attributes and different types of trees' attributes within this plot.

Services: Allow developers to define objects of a plot. From the objects, developers can store, read, or reset attributes about the plot.

Implemented By: Digital Twin Forest

7.1.13 FirstPersonPlayer Module (M13)

Secrets: The algorithms to control first person view movement in the virtual forest.

Services: Allow users to move and navigate through the forest to browse trees, terrain, sky. etc.

Implemented By: Digital Twin Forest

7.1.14 JsonFile Module (M14)

Secrets: Data structure to for storing environment data and tree parameters.

Services: Json files are used to store data for our system. There are modules to read/write data from/to json files in order to display data on the screen and update data.

Implemented By: Digital Twin Forest

7.2 Viewer Modules

7.2.1 MainPageDisplay Module (M15)

Secrets: A GUI which contains background image and different buttons.

Services: Provide users some options at the beginning of the software. Options include starting browsing virtual forest, checking instructions, checking team members' contact information, quitting the software.

Implemented By: Digital Twin Forest and Unity(UI From Unity Engine).

7.2.2 StartButton Module (M16)

Secrets: A GUI of the **Start** button.

Services: Provide a button GUI for users to start browsing virtual forest after clicking it.

Implemented By: Unity(UI From Unity Engine).

7.2.3 InstructionButton Module (M17)

Secrets: A GUI of the **Instruction** button.

Services: Provide a button GUI for users to check software instructions after clicking it.

Implemented By: Unity(UI From Unity Engine).

7.2.4 ContactUsButton Module (M18)

Secrets: A GUI of the **Contact Us** button.

Services: Provide a button GUI for users to check team members' information after clicking it.

Implemented By: Unity(UI From Unity Engine).

7.2.5 QuitButton Module (M19)

Secrets: A GUI of the **Quit** button.

Services: Provide a button GUI for users to quit the software after clicking it.

Implemented By: Unity(UI From Unity Engine).

7.2.6 InstructionInfoDisplay Module (M20)

Secrets: A GUI that contains instructions for the software.

Services: Provide users instructions about how to use the software.

Implemented By: Unity(UI From Unity Engine).

7.2.7 ContactUsInfoDisplay Module (M21)

Secrets: A GUI that contains developers' contact information.

Services: Provide users developers' contact information.

Implemented By: Unity(UI From Unity Engine).

7.2.8 BackButton Module (M22)

Secrets: A GUI of the **Back** button.

Services: Provide a button GUI for users to go back to the main page after clicking it.

Implemented By: Unity(UI From Unity Engine).

7.2.9 UpdateDataDisplay Module (M23)

Secrets: A GUI which contains buttons, drop drown menu, and an input box.

Services: Users can update environmental data and tree parameters from this page.

Implemented By: Unity(UI From Unity Engine).

7.2.10 EnvDataSelectionButton Module (M24)

Secrets: A GUI of the **Update Environmental Data** button.

Services: Provide a button GUI for users to choose to update environmental data after clicking it.

Implemented By: Unity(UI From Unity Engine).

7.2.11 DataTypeSelectionButtons Module (M25)

Secrets: A group of buttons' GUI in the **Update Data** page.

Services: Provide user options to choose what data to update.

Implemented By: Unity(UI From Unity Engine).

7.2.12 NewDataInputBox Module (M26)

Secrets: A GUI of an input box in the **Update Data** page.

Services: Provide users a way of entering updated data.

Implemented By: Unity(UI From Unity Engine).

7.2.13 SaveButton Module (M27)

Secrets: A GUI of the **Save** button.

Services: Provide a button GUI for users to save updated data after clicking it.

Implemented By: Unity(UI From Unity Engine).

7.2.14 CurrentDataDisplay Module (M28)

Secrets: A GUI that contains current selected data. This is in the **Update Data** page.

Services: Provide a way for users to view current selected data.

Implemented By: Unity(UI From Unity Engine).

7.2.15 PlotSelectionDropDown Module (M29)

Secrets: A GUI of drop down menu that contains 15 options. 15 options include options form `plot1` to `plot14` and `all plots`.

Services: Provide a way for users to select a specific plot in order to view/update plot data.

Implemented By: Unity(UI From Unity Engine)..

7.2.16 TreeTypeSelectionDropDown Module (M30)

Secrets: A GUI of drop down menu that contains 7 options. The following are the details

- Red Pine
- Oak
- Beech
- Birch
- White Pine
- Red Maple
- Red Oak

Services: Provide a way for users to select a specific tree type in order to view /update tree parameters.

Implemented By: Unity(UI From Unity Engine).

7.2.17 UpdateDataButton Module (M31)

Secrets: A GUI of the **Update Data** button.

Services: Provide a button GUI for users to go to **Update Data** page after clicking it.

Implemented By: Unity(UI From Unity Engine).

7.2.18 ForestDisplay Module (M32)

Secrets: A GUI that contains the following: Modules

- M1
- M2
- M3
- M22
- M29
- M30
- M33
- M34
- M35
- M36
- M37

Services: Provide users a way to browse virtual forest, check environmental data and tree parameters.

Implemented By: Unity(UI From Unity Engine).

7.2.19 ShowEnvDataButton Module (M33)

Secrets: A GUI of the **Show Environmental Data** button.

Services: Provide a button GUI for users to display/hide environmental data after clicking it.

Implemented By: Unity(UI From Unity Engine).

7.2.20 ShowTreeParamButton Module (M34)

Secrets: A GUI of the **Show Tree Parameters** button.

Services: Provide a button GUI for users to display/hide tree parameters after clicking it.

Implemented By: Unity(UI From Unity Engine).

7.2.21 EnvDataDisplay Module (M35)

Secrets: A GUI that contains environmental data. The following are the details:

- Temperature
- Humility
- Soil C content
- Soil N content
- LAI

Services: Provide a way for users to check environmental data.

Implemented By: Unity(UI From Unity Engine).

7.2.22 TreeParamDisplay Module (M36)

Secrets: GUI that contains tree parameters. The following are the details:

- DHB
- Density
- Height
- Age

Services: Provide a way for users to check tree parameters.

Implemented By: Unity(UI From Unity Engine).

7.2.23 PauseIndicatorDisplay Module (M37)

Secrets: A GUI that contains information to tell if the system is paused.

Services: Provide users a way to tell if the system is paused.

Implemented By: Unity(UI From Unity Engine).

7.3 Controller Modules

7.3.1 JsonFileReader Module (M38)

Secrets: Algorithms to read data from Json files.

Services: Provide service of reading data from Json files and store them into corresponding objects. When users click **Show Environmental Data** button and **Show Tree Parameters** button, this module will be executed. Then, data will be read from corresponding Json files, stored in corresponding objects and waiting for future processing.

Implemented By: Digital Twin Forest

7.3.2 JsonFileWriter Module (M39)

Secrets: Algorithms to write data to Json files.

Services: Provide service of reading data from the input box and write them into corresponding files. When users click **Save** button in **Update Data** page, this module will be executed.

Implemented By: Digital Twin Forest

7.3.3 PauseManager Module (M40)

Secrets: Algorithm to control the time flow of the software.

Services: When users want to check the environmental data or tree parameters, they have the option to pause the system. When the system is paused, mouse and first person view are disconnected, therefore, users can feel more convenient to click on different buttons to display different information.

Implemented By: Digital Twin Forest

7.3.4 PlayerMovement Module (M41)

Secrets: Algorithms to control the first person view movement.

Services: Allow users to move in the forest by using keyboards `W, S, A, D` and adjust view angle by using the mouse.

Implemented By: Digital Twin Forest

7.3.5 NewDataInputBoxController Module (M42)

Secrets: Algorithm of accepting user inputs from the input box.

Services: This module will be responsible for accepting inputs and verify that inputs are appropriate(For our software, we need to make sure that users only enter integers or decimal numbers). After that, new entered data will be stored in this module.

Implemented By: Digital Twin Forest

7.3.6 StartButtonController Module (M43)

Secrets: The operations of displaying virtual forest after users click the **Start** button.

Services: Change scene to forest after clicking **Start** button.

Implemented By: Digital Twin Forest and Unity(Scene Manager)

7.3.7 InstructionButtonController Module (M44)

Secrets: The operations of displaying instructions after users click the **Instruction** button.

Services: The screen will change from main page to instruction page after clicking **Instruction** button.

Implemented By: Digital Twin Forest

7.3.8 UpdateDataButtonController Module (M45)

Secrets: The operations of displaying update data page after users click the **Update Data** button.

Services: The screen will change from main page to Update data page after clicking **Update Data** button.

Implemented By: Digital Twin Forest

7.3.9 ContactUsButtonController Module (M46)

Secrets: The operations of displaying developers' contact information after users click the **Contact Us** button.

Services: The screen will change from main page to developers' contact information page after clicking **Contact Us** button.

Implemented By: Digital Twin Forest

7.3.10 QuitButtonController Module (M47)

Secrets: The operations of quitting the software after users click the **Quit** button.

Services: The software will be closed after clicking **Quit** button.

Implemented By: Digital Twin Forest

7.3.11 BackButtonController Module (M48)

Secrets: The operations of going back to the main page after users click the **Back** button.

Services: The screen will show main page after clicking **Back** button.

Implemented By: Digital Twin Forest

7.3.12 PlotSelectionDropDownController Module (M49)

Secrets: The operations of selecting different plots after users click buttons from the **Plot Selection Drop Down Menu**.

Services: By selecting plots, users can check environmental data and tree parameters from different plots.

Implemented By: Digital Twin Forest

7.3.13 TreeTypeSelectionDropDownController Module (M50)

Secrets: The operations of selecting different tree types after users click buttons from the **Tree Type Selection Drop Down Menu**.

Services: By selecting tree types, users can check tree parameters of different types of trees.

Implemented By: Digital Twin Forest

7.3.14 ShowEnvDataButtonController Module (M51)

Secrets: The operations of displaying environmental data after users click the **Show Environmental Data** button.

Services: Environmental data will be displayed on the screen after clicking **Show Environmental Data** button.

Implemented By: Digital Twin Forest

7.3.15 ShowTreeParamButtonController Module (M52)

Secrets: The operations of displaying tree parameters after users click the **Show Tree Parameters** button.

Services: Tree parameters will be displayed on the screen after clicking **Show Tree Parameters** button.

Implemented By: Digital Twin Forest

7.3.16 EnvDataSelectionButtonController Module (M53)

Secrets: The operations of displaying different environmental data types after users click the **Environmental Data** button in **Update Data** page.

Services: The software will display all environmental data types for users to choose in order to update one specific environmental data.

Implemented By: Digital Twin Forest

7.3.17 DataTypeSelectionButtonsController Module (M54)

Secrets: The operations of choosing different data types in order to update data.

Services:

- If users click **Environmental Data** button, the following buttons will be displayed:
  - Humility
  - Temperature
  - Soil C content
  - Soil N content
  - LAI
- If users select a specific type of tree, the following buttons will be displayed:
  - Density
  - Age

- Height
- DBH

Implemented By: Digital Twin Forest

7.3.18 SaveButtonController Module (M55)

Secrets: The operations of saving newly entered data after users click the **Save** button in **Update Data** page.

Services: This module will first access newly entered data from M42, and then call corresponding methods from M39 to store data in Json files.

Implemented By: Digital Twin Forest

8 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes.

Table 5: Trace Between Functional Requirements and Modules

Functional Req.	**Modules**
FR1	M15, M17, M20, M44
FR2	M15, M16, M43
FR3	M15, M16, M43
FR4	None(see notes below)
FR5	M1, M2, M3, M32
FR6	M29, M30, M33, M34
FR7	M4, M5, M6, M7, M8, M9, M10, M11, M12, M14, M29, M30, M33, M34, M35, M36, M38
FR8	None(see notes below)
FR9	M37, M40
FR10	M13, M41
FR11	M13, M41
FR12	M35
FR13	M35
FR14	M22, M48
FR15	M19, M47
FR16	M14, M23, M24, M25, M26, M27, M28, M29, M30, M31, M39, M42, M45, M49, M50, M53, M54, M55
FR17	M17, M20, M44

Some notes about FR4 and FR8

- For FR4, the requirement is the following:
  The product shall display the percentage of progress while the forest models are being loaded.
  Rationale: During the development, we found out that forest models can be loaded pretty fast, therefore, this requirement can be removed.

- For FR8, the requirement is the following:
  The product shall allow users to select a specific tree in order to view related information (Tree Parameters).

Rationale: During the development, we found out that this requirement is not attainable. The reason is that it is impossible to measure all the trees in the forest. Instead, after discussing with out stakeholders, we decided the use panels to display tree parameters for different types of trees. In order to display tree parameters of different types of trees via panels, we need M14, M30, M34, M36, M38, M50, M52.

Table 6: Trace Between Non-Functional Requirements and Modules (First Table)

Non-Functional Req.	**Modules**
LF 1.1	All the modules (Please check Table 1, 2, 3, and 4).
LF 1.2	M1, M2, M3
LF 2.1	M1, M2, M3
LF 2.2	All the Viewer modules (Please check Table 2, 3).
UH 1.1	M17, M20, M44
UH 2.1	All the Viewer modules (Please check Table 2, 3).
UH 3.1	M17, M20, M44
UH 4.1	M16, M17, M18, M19, M22, M24, M25, M26, M27, M29, M30, M31, M33, M34
UH 4.2	M16, M17, M18, M19, M22, M24, M25, M26, M27, M29, M30, M31, M33, M34
UH 5.1	M40, M41, M42, M43, M44, M45, M46, M47, M48, M49, M50, M51, M52, M53, M54, M55
UH 5.2	All the Viewer modules (Please check Table 2, 3).
PR 1.1	All the Controller modules (Please check Table 4).
PR 1.2	M1, M2, M3 and All the Viewer modules (Please check Table 2, 3).
PR 1.3	M1, M2, M3, M43
PR 3.1	M14
PR 4.1	M4, M5, M6, M7, M8, M9, M10, M11, M12, M14, M43
PR 4.2	All the Controller modules (Please check Table 4).
PR 5.1	All the Controller modules (Please check Table 4).
PR 6.1	M1, M2, M3
PR 7.1	N/A (Our MVC Design can be helpful for this requirement.)
PR 8.1	N/A (This depends on our maintenance work.)
OE 1.1	N/A (Unity Game Engine helps to achieve this goal.)
OE 1.2	M40, M41, M42, M43, M44, M45, M46, M47, M48, M49, M50, M51, M52, M53, M54, M55
OE 2.1	N/A (Unity Game Engine helps to achieve this goal.)
OE 3.1	N/A (Unity Game Engine helps to achieve this goal.)
OE 4.1	N/A (This depends on our maintenance work.)

Table 7: Trace Between Non-Functional Requirements and Modules (Second Table)

Non-Functional Req.	**Modules**
OE 4.2	All the Controller modules (Please check Table 4).
MS 1.1	N/A (This is not related to software design.)
MS 1.2	N/A (This is not related to software design.)
MS 1.3	N/A (This is not related to software design.)
MS 2.1	M18, M21, M46
MS 3.1	N/A (Unity Game Engine helps to achieve this goal.)
MS 3.2	N/A (This is not related to software design.)
SR 1.1	N/A (This is not related to software design.)
SR 2.1	All the Controller modules (Please check Table 4).
SR 2.2	All the Controller modules (Please check Table 4).
SR 2.3	M42
SR 2.4	M38
SR 2.5	M35, M36
SR 2.6	M14
SR 3.1	All the modules (Please check Table 1, 2, 3, and 4).
SR 3.2	All the modules (Please check Table 1, 2, 3, and 4).
CP 1.1	All the Viewer modules (Please check Table 2, 3).
LR 2.1	All the Viewer modules (Please check Table 2, 3).

Table 8: Trace Between Anticipated Changes and Modules

AC	**Modules**
AC1	M15, M16, M17, M18, M19 , M31
AC2	M35, M36
AC3	M14
AC4	M4, M5, M6, M7, M8, M9, M10, M12, M14, M38, M39
AC5	M11, M12, M14, M38, M39

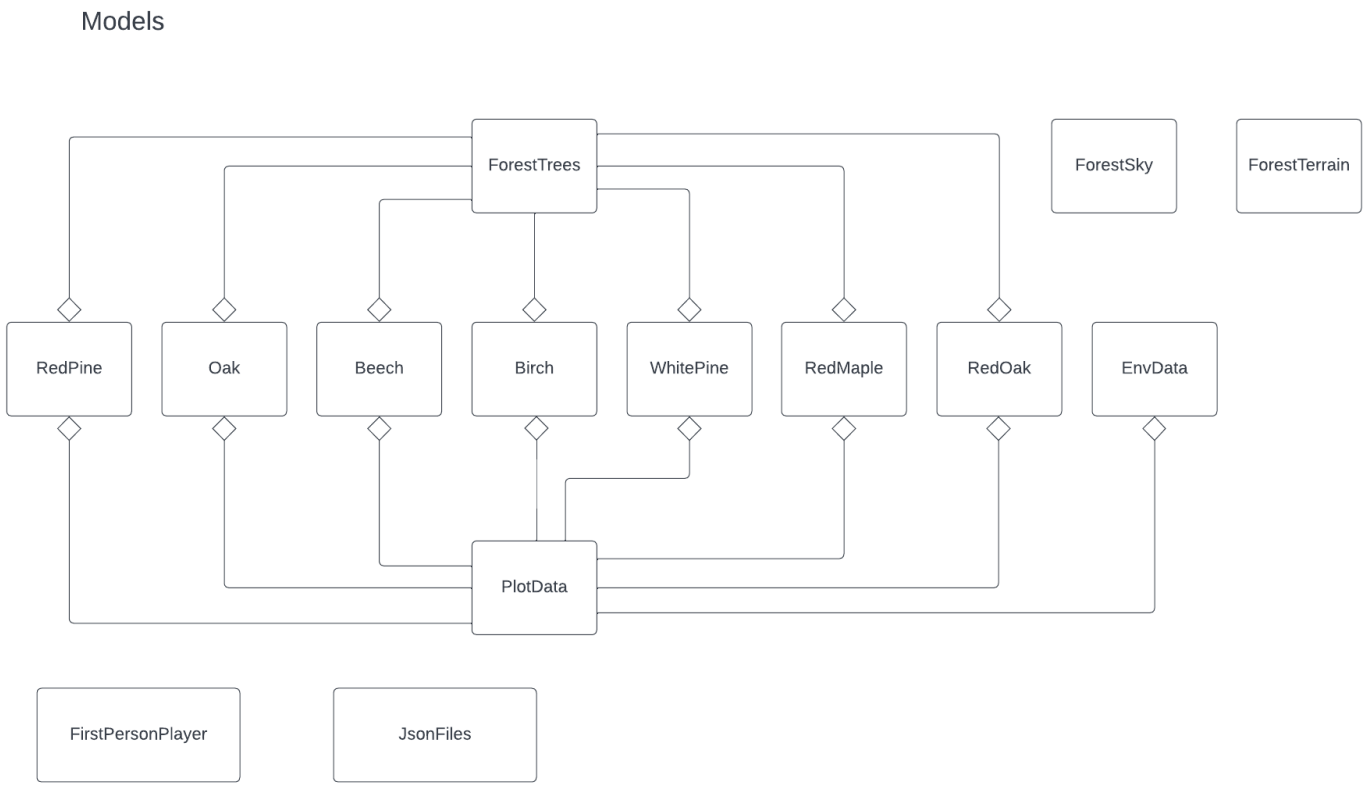
9 Relationship Between Modules

This section presents relationship between modules. The contents are organized as the following:

- Section 9.1 presents relationships within different Model modules.
- Section 9.2 presents relationships within different Viewer modules.
- Section 9.3 presents relationships within different Controller modules.
- Section 9.4 presents relationships between Model modules, Viewer modules, and Controller modules.

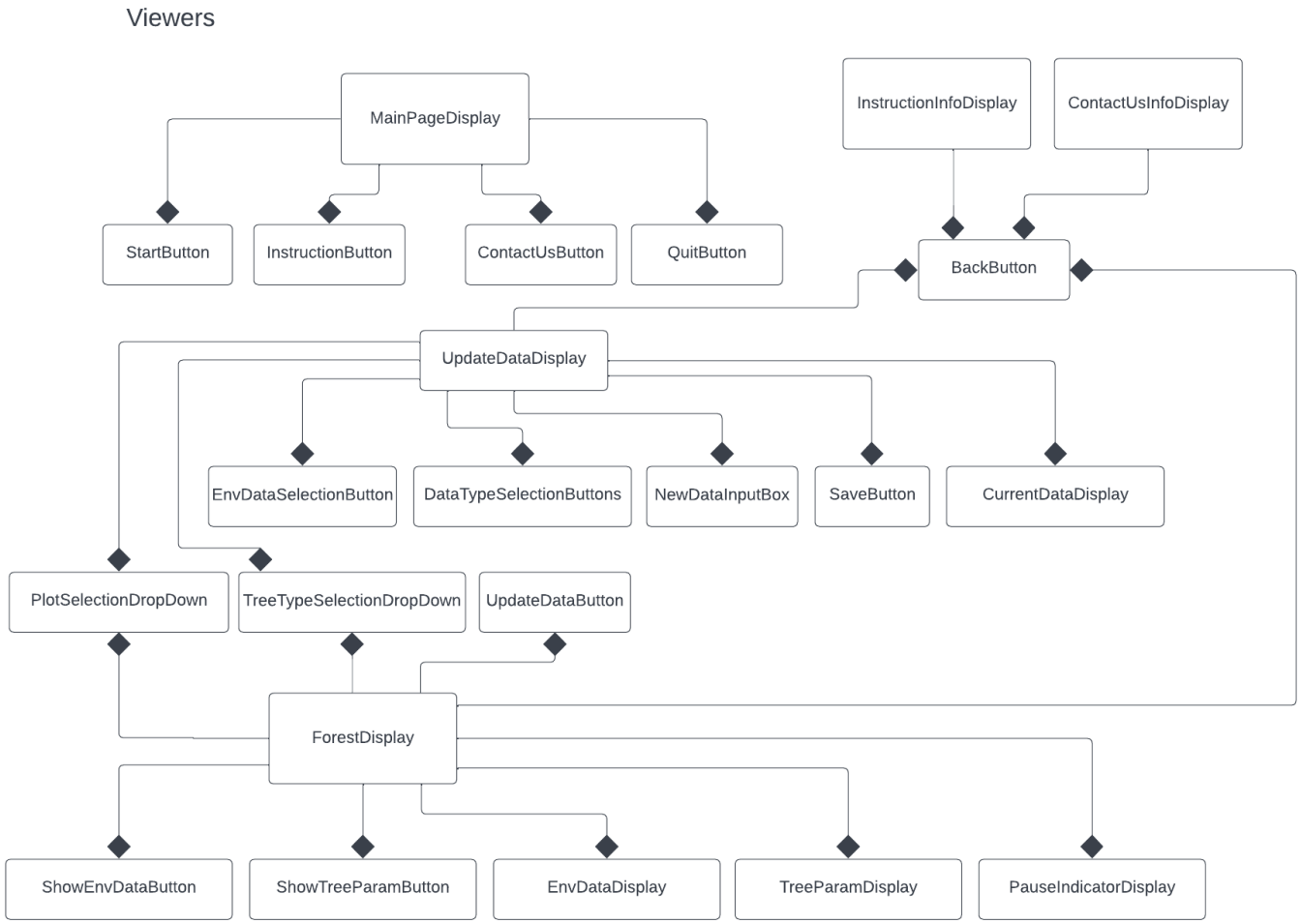
9.1 Relationship within Model Modules

Figure 1: Relationships within Model modules



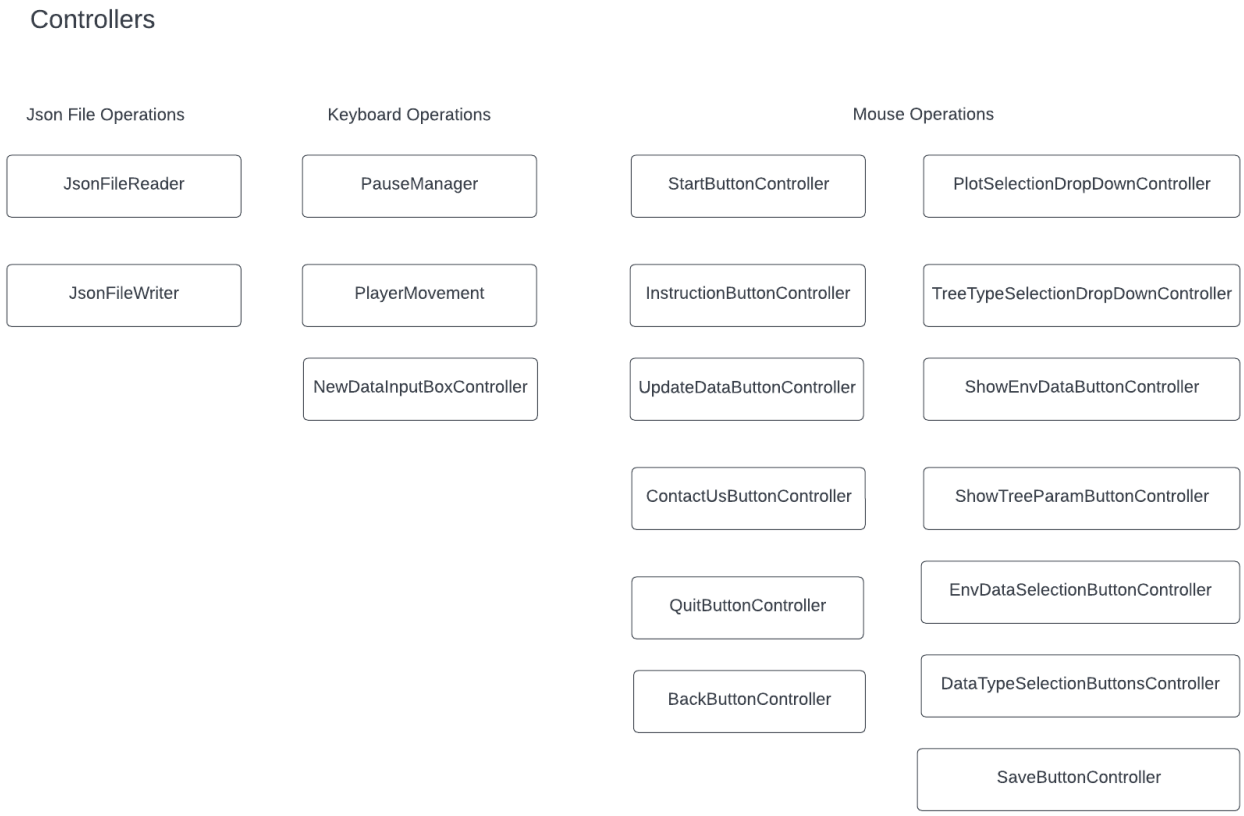
9.2 Relationship within Viewer Modules

Figure 2: Relationships within Viewer modules



9.3 Relationship within Controller Modules

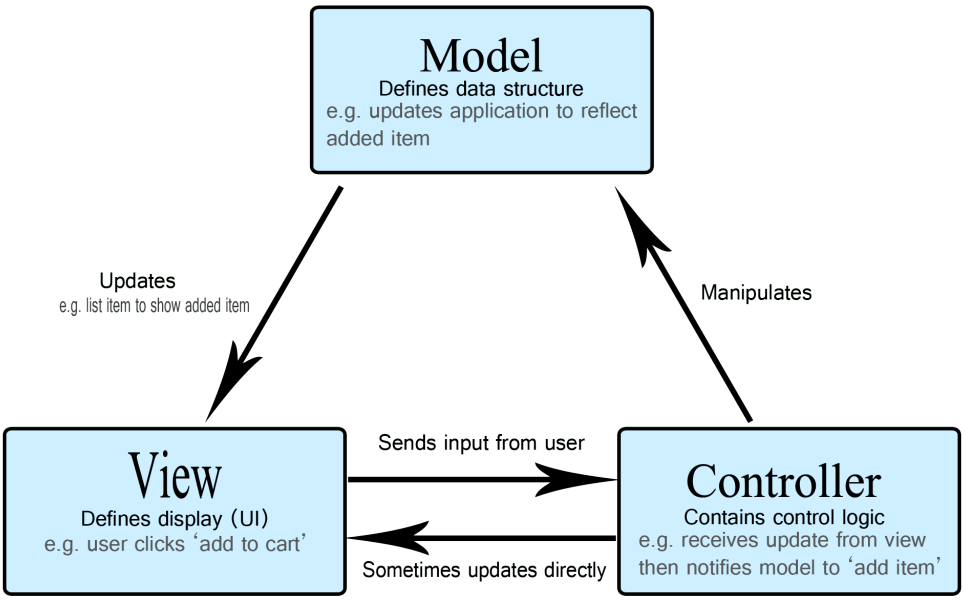
Figure 3: Relationships within Controller modules



9.4 Relationship between Model Modules, Viewer Modules, and Controller Modules

The following is a diagram to show the relationship between models, viewers, and controllers of the MVC architecture.

Figure 4: MVC Architecture



Reference: Click here.
Drawing graphs to represent relationships between Model modules, Viewer modules, and Controller modules is too complicated since we have many modules. Therefore, we chose to use words to describe relationships between between Model modules, Viewer modules, and Controller modules.

9.4.1 Model Updates View

- M12 **Updates** M35
- M12 **Updates** M36
- M1 **Updates** M32
- M2 **Updates** M32
- M3 **Updates** M32

9.4.2 Controller Manipulates Model

- M39 **Manipulates** M14
- M41 **Manipulates** M13
- M55 **Manipulates** M12

9.4.3 View Sends input from user to Controller

- M16 **Sends input from user to** M43
- M17 **Sends input from user to** M44
- M18 **Sends input from user to** M46
- M22 **Sends input from user to** M48
- M24 **Sends input from user to** M53
- M25 **Sends input from user to** M54
- M27 **Sends input from user to** M55
- M29 **Sends input from user to** M49
- M30 **Sends input from user to** M50
- M31 **Sends input from user to** M45
- M33 **Sends input from user to** M51
- M34 **Sends input from user to** M52

9.4.4 Controller Updates View

- M40 **Updates** M37
- M43 **Updates** M32
- M44 **Updates** M20
- M46 **Updates** M21
- M51 **Updates** M35
- M52 **Updates** M36
- M53 **Updates** M25
- M54 **Updates** M25

References

David L. Parnas. On the criteria to be used in decomposing systems into modules. *Comm. ACM*, 15(2):1053–1058, December 1972.

D.L. Parnas, P.C. Clement, and D. M. Weiss. The modular structure of complex systems. In *International Conference on Software Engineering*, pages 408–419, 1984.