EECS3311-W20 — Project Report

Submitted electronically by:

|  |  |  |  |
| --- | --- | --- | --- |
| **Team members** | **Name** | **Prism Login** | **Signature** |
| Member 1: | Hongyuan Hua | hongyuan |  |
| Member 2: | Yonghao Lu | lu1997 |  |
| \*Submitted under Prism account: | | lu1997 |  |

Table of Contents

[1. Requirements for Project SimOdyssey2 2](#_Toc36871668)

[2. BON class diagram overview (architecture of the design) 3](#_Toc36871669)

[3. Table of modules ---- responsibilities and information hiding 5](#_Toc36871670)

[4. Expanded description of design decisions 7](#_Toc36871671)

[5. Significant Contracts (Correctness) 8](#_Toc36871672)

[6. Summary of Testing Procedures 9](#_Toc36871673)

[7. Appendix (Contract view of all classes) 11](#_Toc36871674)

# Requirements for Project SimOdyssey2

Our customer provided us with the following statement of their needs:

The system to be modeled and implemented is a simplified simulation of a galaxy. A two-dimensional grid of sectors represents the galaxy. In the current game you are being asked to program, the size of the grid is 5 by 5. Each sector in the grid is identified by its coordinates in terms of the row number and the column number.

For the purpose of moving the explorer, if the explorer is in any sector, it can travel to any of the 8 adjacent sectors normally. These are found in the north, north-east, east, south-east, south, south-west, west, and north-west positions directly adjacent to the given sector For example, looking at sector (3,3), we can see that (2,4) is the north-east neighbour.

Also, for different kinds of entities, they have different attributes such as some of them are movable and some are not. They also have different functions as well for example MALEVOLENT is going to attack EXPLORER whereas BENGIN is going to defend EXPLORER and kill MALEVOLENT.

The mission of explorer is going to land at sector that with PLANET and YELLOW\_DWARF in order to find a life. AND, of course, avoid death as well.

Errors detection must be done in every execution. We can refer to *simodyysey2-messages.txt* to all the possible messages that may happen and should be the same as oracle.exe.

See *simodyssey2.definitions.txt* for the grammar of the user interface. The acceptance tests at001.expected.txt, at002.expected.txt and at003.expected.txt describe some of the input-output behavior at the console for this project.

# BON class diagram overview (architecture of the design)

A close up of a map

Description automatically generated

The main design concern in this project is to make it simple, organized. This way can make the program easy to modify and make upgrades. Also, most importantly, it makes much easier for us to debug.

The most shinning part in our program which makes our program stands out is that we created a separate Class called ERROR\_DETECTOR to keep tracking of the errors. This class can not only detect the errors but also it can find out the error types. If there is an error, the class ETF\_MODEL is going to call class ERROR\_MESSAGE to throw the error messages. If not, ETF\_MODEL is going to call class MAIN\_MESSAGE to throw messages. When a death happens, ETF\_MODEL is going to call class DEATH\_MESSAGE to throw out the death messages.

The Class ENTITY is the superclass of two different types of entities in this game. One is called MOVABLE which includes entity classes EXPLORER, PLANET, ASTEROID, JANITAUR, MALEVOLENT and BENGIN. Another one is called STATIONARY which includes STARS like BLUE\_GAINT and YELLOW\_DWARF and BLACKHOLE. The reason of doing such kind of iteration is that it makes simple and straightforward to defines different kinds of entities, also, enables us to reuse codes which saves us some extra work.

The Class GALAXY works like a display board which lets the different SECTORs to show up.

The Class SECTOR also works like a board or container of different kinds of ENTITY which can includes up to 4 item.

The Class SHARED\_INFORMATION and RANDOM\_GENERATOR provides SECTOR enough information to generate different ENTITY. The RANDOM\_GENERATOR is going to provide random numbers for movable items except EXPLORER to move.

# Table of modules ---- responsibilities and information hiding

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | ENTITY | **Responsibility**: superclass of entities | **Alternative**: none |
| Abstract | **Secret**: none |

|  |  |  |  |
| --- | --- | --- | --- |
| 1.1 | MOVABLE | **Responsibility**: superclass of movable entities | **Alternative**: none |
| Abstract | **Secret**: none |

|  |  |  |  |
| --- | --- | --- | --- |
| 1.2 | STATIONARY | **Responsibility**: superclass of stationary entities | **Alternative**: none |
| Abstract | **Secret**: none |

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | ERROR\_DETECTOR | **Responsibility**: detects errors and types | **Alternative**: none |
| Concrete | **Secret**: none |

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | MAIN\_MESSAGE | **Responsibility**: output regular messages | **Alternative**: none |
| Concrete | **Secret**: none |

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | DEATH\_MESSAGE | **Responsibility**: output death messages | **Alternative**: none |
| Concrete | **Secret**: none |

|  |  |  |  |
| --- | --- | --- | --- |
| 5 | ERROR\_MESSAGE | **Responsibility**: output error messages | **Alternative**: none |
| Concrete | **Secret**: none |

|  |  |  |  |
| --- | --- | --- | --- |
| 6 | GALAXY | **Responsibility**: create a board for sectors to display | **Alternative**: none |
| Concrete | **Secret**: none |

|  |  |  |  |
| --- | --- | --- | --- |
| 7 | SECTOR | **Responsibility**: create a sector for entities to display | **Alternative**: none |
| Concrete | **Secret**: none |

|  |  |  |  |
| --- | --- | --- | --- |
| 8 | RANDOM\_GENERATOR | **Responsibility**: generate random numbers to let entities move | **Alternative**: none |
| Concrete | **Secret**: none |

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | SHARED\_INFORMATION | **Responsibility**: share some integers to SECTOR for it to create entities | **Alternative**: none |
| Concrete | **Secret**: none |

# Expanded description of design decisions

As what I described before, the most important module in my design is the message detect sections which includes several classes -- ERROR\_DETECTOR, MAIN\_MESSAGE, ERROR\_MESSAGE and DEATH\_MESSAGE.

This section is most important in my design because message output part is very complex and very easy to make a mistake. One single error message can be caused by different kinds of condition. If we just simply put this message detection part in the model, it is very easy for the model class to get messy and bugs. In this case, I decide to handle this from a different approach, by creating class ERROR\_DETECTOR to detect errors.

If there is an error occur in the program, the ERROR\_DETECTOR is going to let the model know the type of the error and then model is going to retrieve corresponding message from ERROR\_MESSAGE class.

If there is not an error occur in the program, the model will simply retrieve corresponding message from MAIN\_MESSAGE class and show it to our clients.

If there is a death occur, the model is also going the retrieve corresponding message from DEATH\_MESSAGE class and show appropriate death message.

By using this message detection design, any corresponding message can be easily modified or upgraded. Also, the it makes model less messy and easier to maintain.

# Significant Contracts (Correctness)

The most significant contracts occur in MOVABLE module. In this project, there are several movable entities and either of them have different attributes and functions such as MALEVOLENT is going to attack EXPLORER whereas BENGIN is going to defend EXPLORER and kill MALEVOLENT. In order to make the MOVABLE class easy to maintain and, also, to keep each entity separately, I made the MOVABLE class abstracted which all its sub-entities are going to define their own unique functions.

Also, when clients are going to call the entities, they don’t need to know the entities’ types. Clients just need to call the MOVABLE. My design is going to cast the MOVABLE to the corresponding type.

This iteration design approach makes the program readable, easy to modify and upgrades. Also, the most importantly, it successfully filled the requirements of the contract and made the design stands out.

# Summary of Testing Procedures

|  |  |  |
| --- | --- | --- |
| **Test file** | **Description** | **Passed** |
| t1.txt | play in play mode and try out all commands in basic situation | ✔ |
| t2.txt | play in test mode and see if main massage prints out correctly | ✔ |
| t3.txt | play in play mode and see if error massage prints out as expected | ✔ |
| t4.txt | play in test mode and see if error massage and death message work out as expected | ✔ |
| t5.txt | play in test mode and try be alive for long to see if order of death is correct | ✔ |
| t6.txt | play in test with fixed number to see the reproduce function work correctly | ✔ |
| t7.txt | move into full, move into blackhole, move into sector with asteroid | ✔ |
| t8.txt | liftoff after a liftoff or wormhole | ✔ |
| t9.txt | wormhole, status, land | ✔ |
| t10.txt | win a game and abort | ✔ |
| t11.txt | play game in test mode with fixed number and try to die for three time | ✔ |
| t12.txt | wormhole after land, wormhole and pass | ✔ |
| t13.txt | multiple asteroid tries to kill explorer, see if correct order is maintained | ✔ |
| t14.txt | multiple janitor tries to kill asteroid, see if correct order is maintained | ✔ |
| t15.txt | play in test mode with fixed number, see if Malevolent attack explorer, and Benign work correctly | ✔ |
| t16.txt | play multiple game to and die halfway to see if death message work correctly | ✔ |
| t17.txt | play in test mode with fixed number, check the functionality of Malevolent attacking, status and death message | ✔ |
| t18.txt | test with incorrect order of numbers at: in game, not in game, after abort, after death and at start of program | ✔ |
| t19.txt | play and test and play with winning the game | ✔ |
| t20.txt | play in test mode with large fixed number, pass for many times and see the death, reproduce and order of them | ✔ |
| upboundary0.txt ~  upboundary19.txt | I also wrote a java program to generate random commands to try out all possible case, each file has 1000 random commands. | ✔ |

# Appendix (Contract view of all classes)

手机屏幕截图

描述已自动生成

手机截图图社交软件的信息

描述已自动生成

手机屏幕截图

描述已自动生成

手机屏幕截图

描述已自动生成

手机截图图社交软件的信息

描述已自动生成

手机屏幕截图

描述已自动生成

手机截图图社交软件的信息

描述已自动生成

手机屏幕截图

描述已自动生成

手机屏幕截图

描述已自动生成

手机屏幕截图

描述已自动生成

手机屏幕截图

描述已自动生成

手机屏幕截图

描述已自动生成

手机屏幕截图

描述已自动生成

手机截图图社交软件的信息

描述已自动生成

手机屏幕截图

描述已自动生成

手机屏幕截图

描述已自动生成



手机屏幕截图

描述已自动生成

手机屏幕截图

描述已自动生成