Asteroid Mining

Team name: Pied Pipers

Title: Prototype Program

Supervisor:

Dr. Balla Katalin

Members:

Abdelrahman Desoki	DOHDZF	adesoki@edu.bme.hu
Neda Radonjic	NQC17Y	neda.radonjic@edu.bme.hu
Tushig Bat-Erdene	QBI3JH	bat-erdene.tushig@edu.bme.hu
Chaitanya Arora	BWTFX8	chaitanya.arora@edu.bme.hu
Janibyek Bolatkhan	BOI6FK	janibyekbolatkhan@edu.bme.hu
Kasay Ito	LX5RFB	kasay.ito@edu.bme.hu

May 2 , 2022

User interface specification

11.1 Graphical User Interface

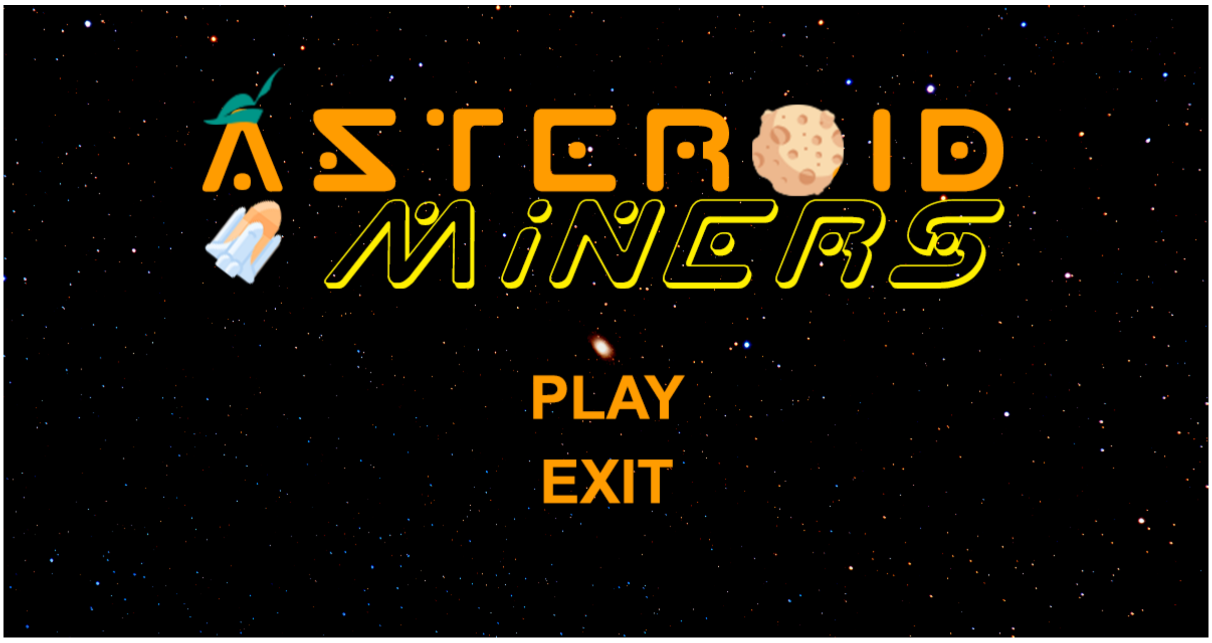


Figure 1. Introduction screen of the game

Figure 1 displaying the first screen of the game when it is successfully executed or compiled. The “PLAY” button is responsible for starting the game and the “STOP” button is responsible for terminating the game.

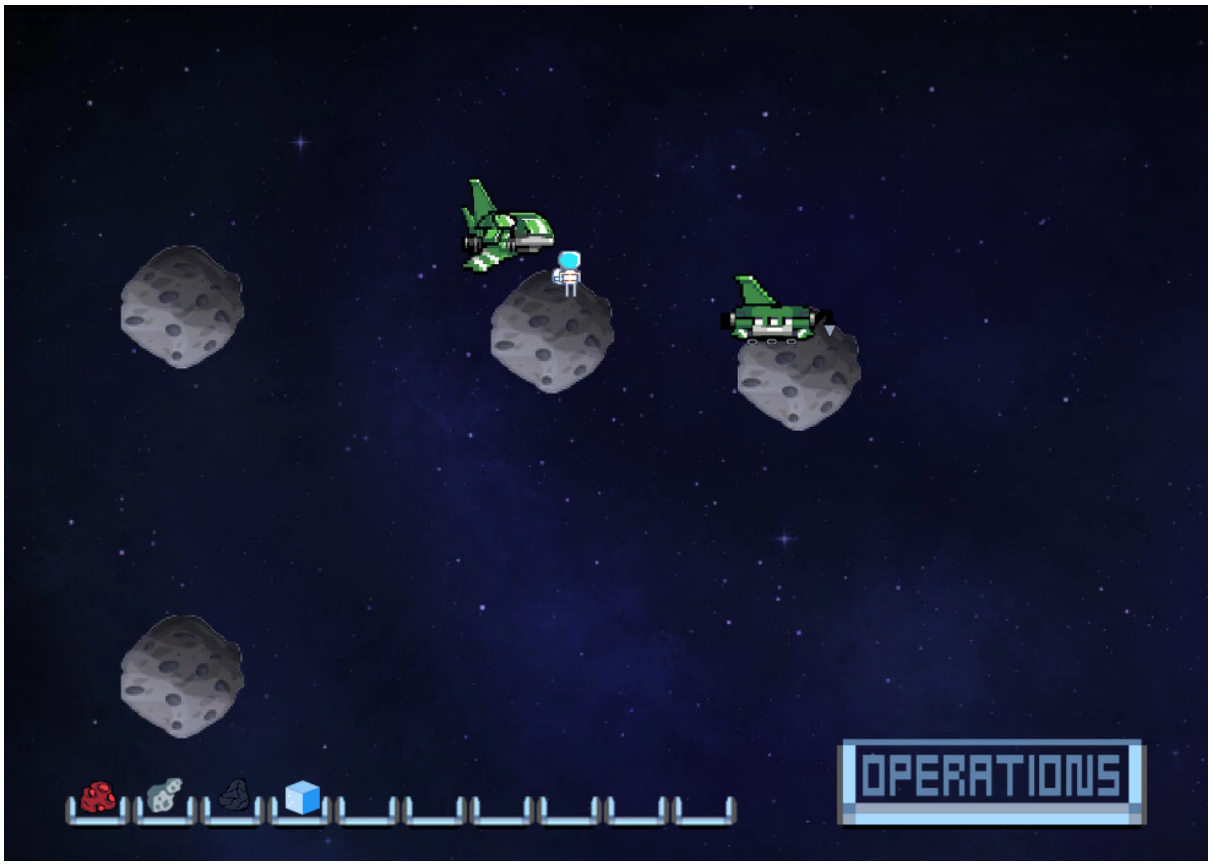


Figure 2. Game scenario

Figure 2 is demonstrating one of the scenarios of the game. Player can see the storage from the left bottom part of the game screen which is displaying the resources that the player gathered during the game. The “Operations” button is responsible for screening the information of the important keys for actions such as drilling, mining, moving and so on.

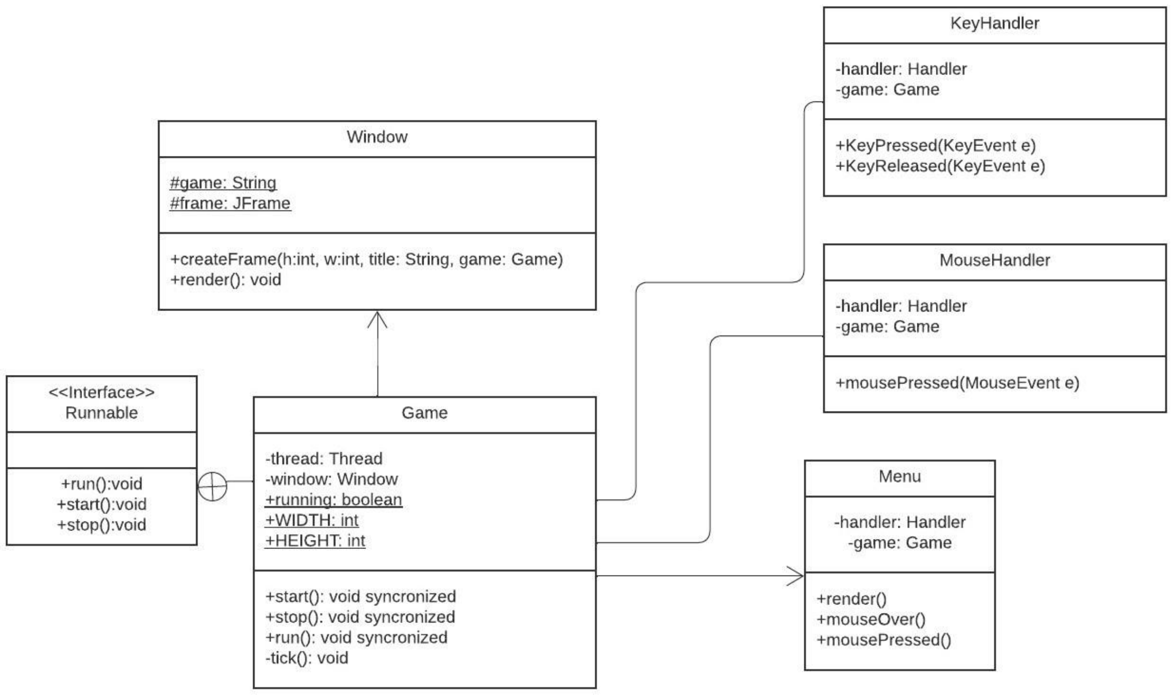
11.2 Architecture of the graphical system

11.2.1 Principles of the GUI

Our solution is based on simple Model-View-Controller (MVC) model.

- Model is considered as class Game and class Handler as abstraction, namely, these two classes can pass information from Controllers to Views although the authentic data is stored in other objects (e.g. Settelr, Asteroid).
- Controllers (i.e. KeyHandler, MouseHandler) can access Model only through class Game or class Handler.
- View is comprised of a class Window for game itself and a class Menu for menu in the game.

11.2.2 GUI Structure diagram



11.3 GUI Classes

We abbreviate some functions which come from super class or interfaces since they do not relate to necessary and important logic behind the UI or GUI specification.

11.3.1 Window

·Responsibility:

The Window class's responsibility is to create the JFrame window. This class outputs according to the computed data by Model. This is one of View in MVC model.

Attributes

- **#game: String:** distinguishes windows in case there are multiple windows
- **#frame:JFrame:** Jframe of the window class

Methods

- **public void createFrame(h:int, w:int, title: String, game: Game):** it sets the dimensions settings of the given frame with corresponding parameters
- **+render():void -** renders (repaints) the actual graphics of the game. For example, frame-per-second is determined in Model, however, this class repaints the screen actually.

11.3.2 Game

· **Responsibility:**

This class is the main class of the program. It is responsible of starting(initializing) and ending the program as well as containing important handling classes and all existing objects. For the User interface of the program, it is extending the built in class Canvas to be able to be drawn and rendered. And implementing the built in Runnable interface to be run as a multi-threaded program. Please refer back to the past class diagram for more precise responsibility. If there are corresponding events notified by Controllers or Views, this has the responsibility for defining which functions are callbacks. This is one of Model in MVC model.

· **Superclasses**

- **Game inherits from Canvas.**

· **Interfaces**

- **<<Interface>> Runnable**

· **Attributes**

- **-thread: Thread -** in order to fetch the signal and update information, it uses a thread
- **-window: Window** - the game window
- **+running:boolean -** representes the current state
- **+WIDTH:int -** indicates the width of the game
- **+HEIGHT:int -** indicates the height of the game

· **Methods**

- **+start():void synchronized -** Creates a new Thread.
- **+stop():void synchronized -** Stops the Thread. This can be used for pausing the game, therefore, we do not specify the dynamic behaviour as a sequence diagram, but we keep this remained for further extendability.
- **+run():void synchronized -** runs the program by setting frame per second as well as executing the methods render and tick method.
- **-tick():void -** it calls the tick methods of the all objects which are changeable. For example, this method is used by the Visitor class the set the location. Therefore, this function can manage time-based behaviour.

11.3.3 Menu

· **Responsibility:**

Menu page window of the game program where user can select the state of game program. Currently, menu has 3 option, such Play, instructions, exit. This class inputs and outputs according to the computed data by Model. This is one of View-Control in MVC model. (because of superclass Canvas)

· **Superclasses**

- **Menu inherits from Canvas.**

· **Attributes**

- **-handler:Handler -** handler of the all the existing objects in the Asteroid belt. This is described in the former class diagram.
- **-game:Game:** game class of the program which is used to accessing the important attributes and objects of the game since it is main class

· **Methods**

- **+render():** renders the graphics of the menu page. For example, frame-per-second is determined in Model, however, this class repaints the screen actually.
- **+mouseOver():** checks if the mouse is on the menu setting
- **+mousePressed():** takes the mouse presses event and checks which menu setting options is pressed and thus changes the state of the game by corresponding chosen option.

11.3.4 KeyHandler

· **Responsibility**

This class has responsibility such that it must handle the event related to key inputs. The behaviour for an input depends on each key. This class notifies Model about events. This is one of Controller in MVC model.

· **Superclasses**

- **KeyHandler inherits from KeyAdapter.**

· **Attributes**

- **-handler:Handler -** handler of the all the existing objects in the Asteroid belt. This is described in the former class diagram.
- **-game:Game:** game class of the program which is used to accessing the important attributes and objects of the game since it is main class

· **Methods**

- **+KeyPressed(KeyEvent e)-** Deals with the events of Direction changing(left, right, up,down), Drilling, Hiding, and the like depending on the key pressed.
- **+KeyReleased(KeyEvent e)-** Deals with the events when the key is released from pressing state.

11.3.5 <<Interface>> Runnable

· **Responsibility:**

It makes the class with thread runnable and gives the ability to override its methods. Since this is a built-in class, we do not explain the details in this documentation.

11.3.6 MouseHandler

· **Responsibility:**

Handles the all the mouse click events of game screen. This class notifies Model about events. This is one of Controller in MVC model.

· **Superclasses**

- **KeyHandler inherits from MouseAdapter.**

· **Attributes**

- **-handler:Handler -** handler of the all the existing objects in the Asteroid belt. This is described in the former class diagram.
- **-game:Game:** game class of the program which is used to accessing the important attributes and objects of the game since it is main class

· **Methods**

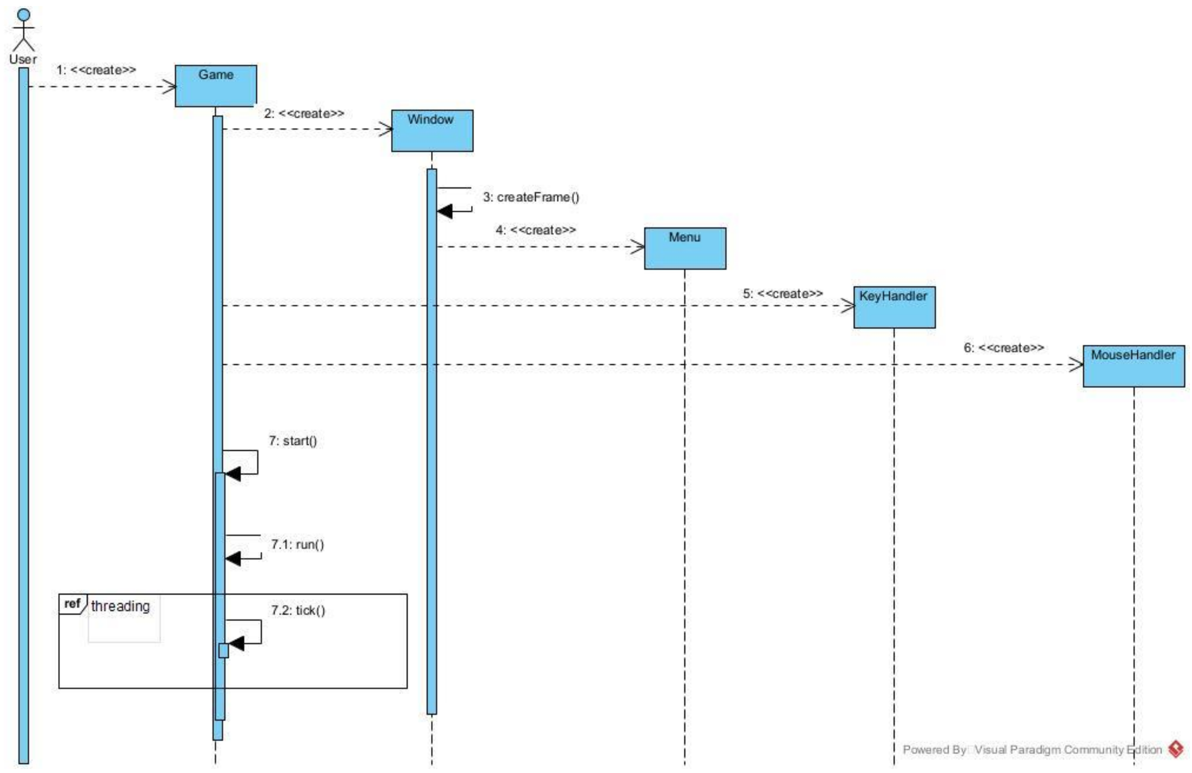
- **+mousePressed(MouseEvent e)-**Gets the coordinates of the mouse click event.

11.4 Dynamic connection between the model and the GUI

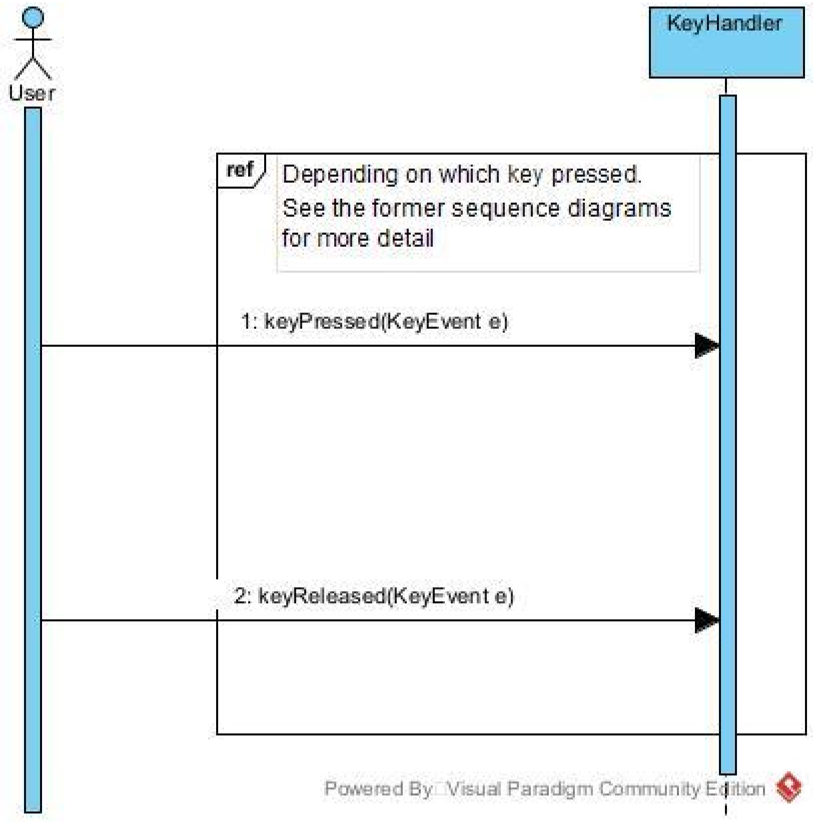
The function “stop()” in class Game is not described in any of sequence diagrams, however, the reason for that is due to the fact that the interface Runnable has stop() and pausing of the game can affect the behaviour of GUI as well.

We do not specify about pausing of a game because it is out of our scope now.

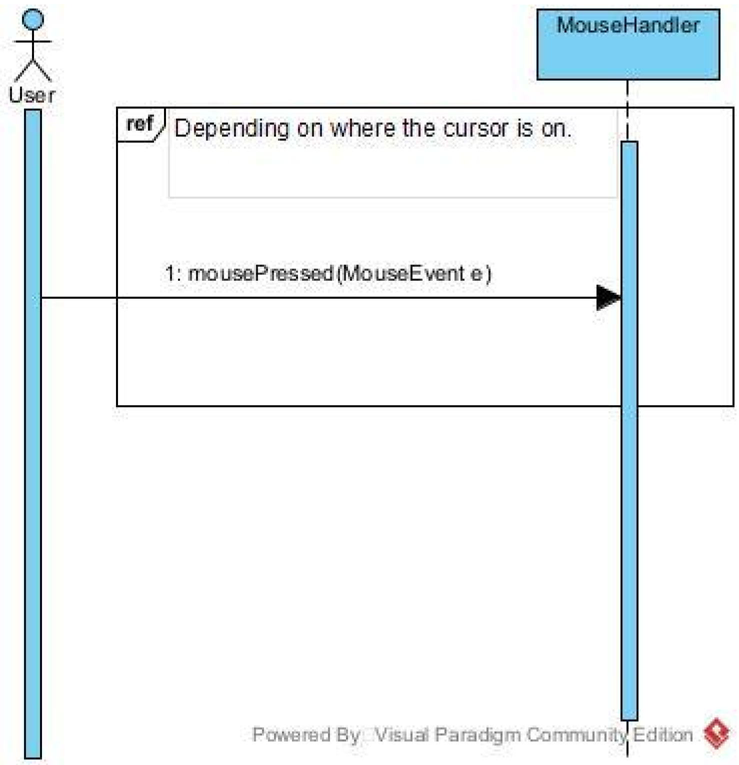
11. 4. 1. initialization



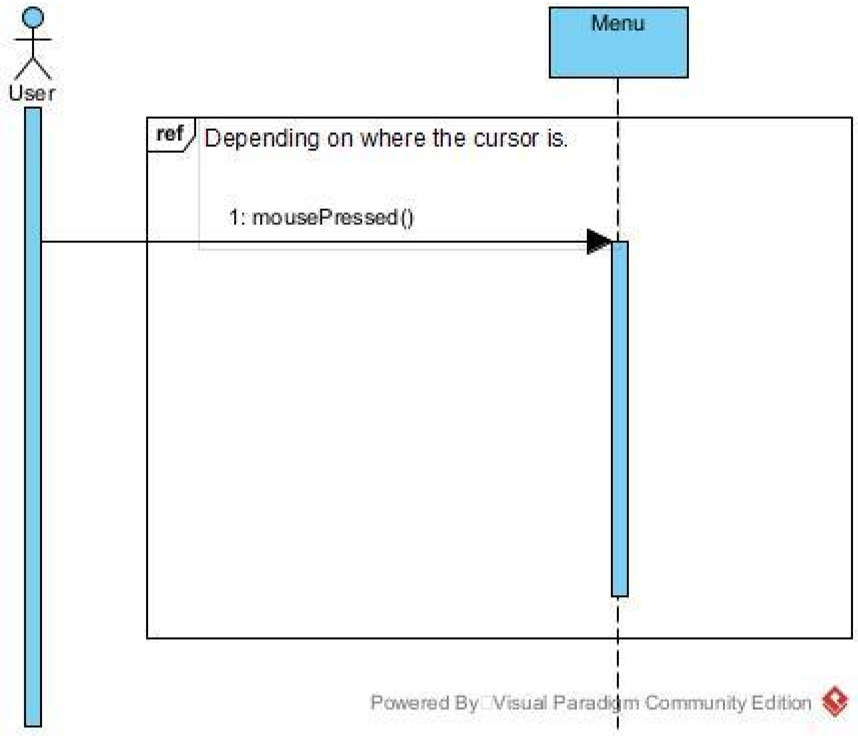
11. 4. 2. control key


User
KeyHandler
ref
Depending on which key pressed. See the former sequence diagrams for more detail
1: keyPressed(KeyEvent e)
2: keyReleased(KeyEvent e)
Powered By: Visual Paradigm Community Edition

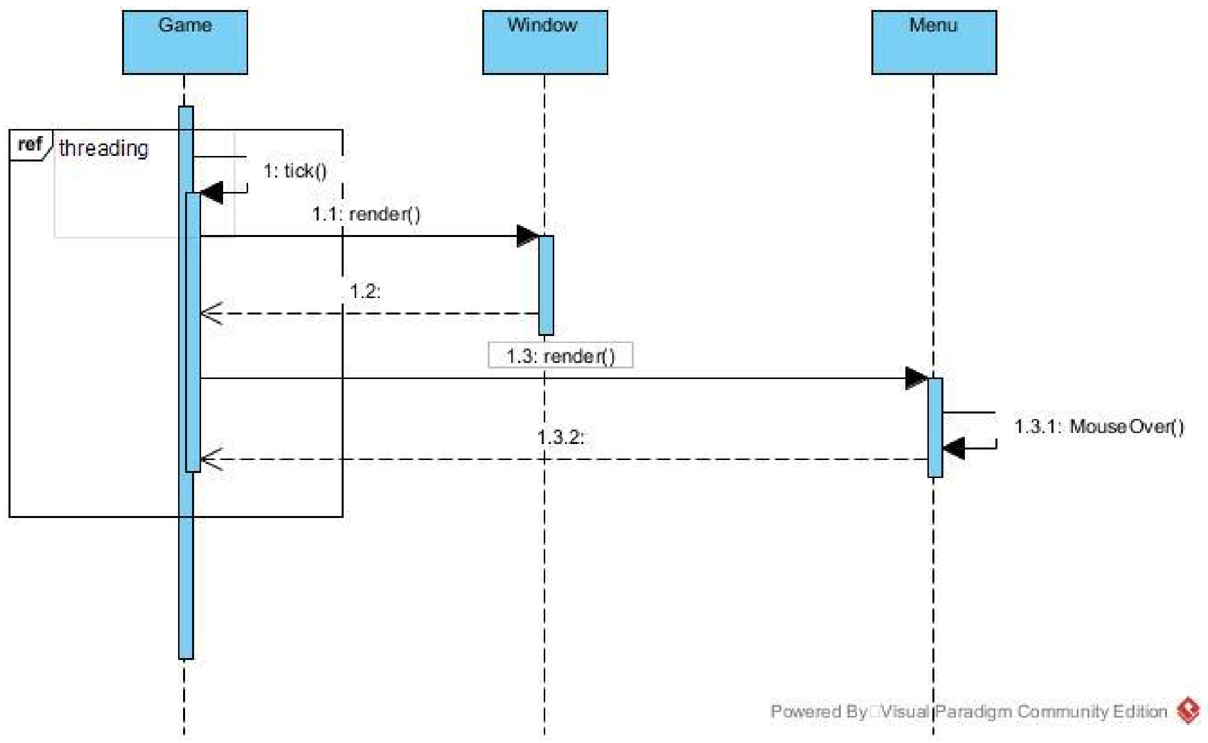
11. 4. 3. control mouse


User
MouseHandler
ref
Depending on where the cursor is on.
1: mousePressed(MouseEvent e)
Powered By Visual Paradigm Community Edition

11. 4. 4. selecting on menu


User
Menu
ref
Depending on where the cursor is.
1: mousePressed()
Powered By Visual Paradigm Community Edition

11. 4. 5. during game



11.5 Protocol

Start (date & time)	Duration (hours)	Performer(s) name	Activity description
29/04/2022	**2.5 hours**	**The whole team**	**Discussing the needed changes, creating the new class structure diagram and class descriptions, and getting started on the sequence diagrams.**
1/05/2022	3 hours	The whole team	Finalizing the GUI structure diagram, completing the class descriptions and sequence diagrams.
2/05/2022	2 hours	The whole team	Finalizing the documentation and discussing the future steps.

02/05/2022	2 hours	Kasay	really finalizing principles of GUI, class diagram, responsibility for each class and sequence diagrams