Annexure – 1

MISSILE COMMAND GAME

A Mini Project-I Report submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

By

1. N. Lasya (19B01A05B9)
2. V. Harshita Sai(19B01A05A2)
3. N. Vanaja (20B05A0511)
4. K. Lovely Srenika (19B01A0583)
5. K. Durga Sreshta (19B01A05A6)

Under the esteemed guidance of
Mr. A. Seenu MTech (PhD)
Associate Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING SHRI VISHNU ENGINEERING COLLEGE FOR WOMEN(A)
(Approved by AICTE, Accredited by NBA & NAAC, Affiliated to JNTU Kakinada)
BHIMAVARAM – 534 202 2020 – 2021

Annexure – 2

SHRI VISHNU ENGINEERING COLLEGE FOR WOMEN (A)

(Approved by AICTE, Accredited by NBA & NAAC, Affiliated to JNTU Kakinada)
BHIMAVARAM – 534 202

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

*This is to certify that the Mini Project-I entitled "**Missile Command Game**", is being submitted by **N. Lasya, V.Harshita Sai, K. Lovely Srenika, K. Durga Sreshta, N. Vanaja** bearing the **Regd. No. 19B01A05B9, 19B01A05A2, 19B01A0583, 19B01A05A6, 20B05A0511** in partial fulfillment of the requirements for the award of the degree of "**Bachelor of Technology** in **Computer Science & Engineering**" is a record of bonafide work carried out by her under my guidance and supervision during the academic year* **2020 – 2021** *and it has been found worthy of acceptance according to the requirements of the university.*

Internal Guide **Head of the Department**

ACKNOWLEDGEMENTS

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant encouragement and guidance has been a source of inspiration throughout the course of this project. We take this opportunity to express our gratitude to all those who have helped us in this project.

We wish to place our deep sense of gratitude to **Sri. K. V. Vishnu Raju, Chairman of SVES**, for his constant support on each and every progressive work of us.

Our deep sense of gratitude and sincere thanks to **Dr. G. Srinivasa Rao, Principal of SVECW** for being a source of an inspirational constant encouragement.

Our deep sense of gratitude and sincere thanks to **Dr. P. Srinivasa Raju, Vice Principal of SVECW** for being a source of an inspirational constant encouragement.

Our deeply indebted and sincere thanks to **Dr. P. Kiran Sree, Head of the Department**, for his valuable advice in completing this project successfully.

Our deep sense of gratitude and sincere thanks to **Mr. A. Seenu, Associate Professor** for his unflinching devotion and valuable suggestion throughout our project.

Project Associates

N. Lasya (19B01A05B9)
V.Harshita Sai (19B01A05A2)
K. Lovely Srenika (19B01A0583)
K. Durga Sreshta (19B01A05A6)
N. Vanaja (20B05A0511)

SHRI VISHNU ENGINEERING COLLEGE FOR WOMEN:: BHIMAVARAM (AUTONOMOUS) DEPARTMENT OF CSE Academic Year:: 2020-21 :: II Year II Semester

B.Tech – MINI PROJECT -I:: ABSTRACT

Name of the Class / Section	II CSE-B		
Batch Number	B03		
Project Domain / Technology	Gaming		
Project Title	Missile Command game		
Guide Name	Mr. A. Seenu		
Students Registered	**Registered Number**	**Student Name**	**Student Signature**
	19B01A05B9	NALLAMALLI LASYA	
	19B01A05A2	VISWANADHAM HARSHITA SAI	
	19B01A0583	KURISETI LOVELY SRENIKA	
	19B01A05A6	KAMANI DURGA SRESHTA	
	20B05A0511	NARIBOINA VANAJA	

Signature of Guide	**Signature of Coordinator**	**Signature of Head of the Department**

Abstract of the Project (In 200 words)

Games are usually undertaken for entertainment or fun, and educational purposes too. Some games improve logical thinking, and some improves concentration. Missile Command is one such game which relaxes, prevents boredom, and improves control over actions of the players.

The main aim of the project is to develop "Missile Command" game. In this game, missiles are shot up from ground to hit falling meteors before they hit the cities. Player has to save the city from meteors strike by launching missiles. In this game, missiles are shot by using mouse control.

The objective of the project is to build a game (Missile Command) that improves concentration levels, improve control over the actions of the player, improves the speed of reactivity to the actions of the player.

The way of playing the game is, by using mouse control, player launches the missile which is against the meteor falling from sky. The game can be implemented in java, by creating some GUI components and by associating event handling to it which recognizes mouse actions and performs actions accordingly.

Existing System (If any) – Features & Drawbacks

Missile Command is a 1980 arcade game developed and published by Atari, Inc. It was designed by Dave Theurer.

The game is played by moving a crosshair across the sky background via a trackball and pressing one of three buttons to launch a counter-missile from the appropriate battery. Counter-missiles explode upon reaching the crosshair, leaving a fireball that persists for several seconds and destroys any enemy missiles that enter it. There are three batteries, each with ten missiles; a missile battery becomes useless when all its missiles are fired, or if the battery is destroyed by enemy fire. The missiles of the central battery fly to their targets at much greater speed; only these missiles can effectively kill a smart bomb at a distance.

The game is staged as a series of levels of increasing difficulty; each level contains a set number of incoming enemy weapons. The weapons attack the six cities, as well as the missile batteries; being struck by an enemy weapon results in the destruction of the city or missile battery.

Proposed System – Features
List of objectives/features that are planned to implement.

The objective of the project is to build a game (Missile Command) that improve control over the actions of the player, improves the speed of reactivity to the actions of the player.

Missile Command is an arcade game and can only be played by a single player. In the game there are many meteors falling from sky which may hit the city, so that, city may get destroyed. So, here the aim of the player is to shoot the meteors using missiles and save the city from destruction.

The game screen contains a city, and it has 3 missile launchers these missile launchers can launch missiles. In the game each missile launcher is provided with only fixed number of missiles, in our game the number is 10. If the 10 missiles are used by the player, then missile launcher becomes inactive. Missile launchers are invulnerable from destruction. If the missile launcher has more than 4 missiles in it, then it is shown in bold, else it is shown with only outline of it.

In the game when player presses on the game screen, a cross hair is displayed (which is the destination of the missile), then missile is launched from nearest possible missile launcher. When missile hits the meteor, explosion occurs.

Here, the main aim of the player is shoot missiles using mouse control provided and save the city from destruction.

Literature Survey(if any)

Software & Hardware Requirements

Software Requirements

Programming Language: Java 1.8
Operating System: Windows 10

Hardware Requirements

Processor: i3
RAM: 4GB
Hard disc: 1TB

Annexure – 3

Contents

9.1 Introduction to Java

1.INTRODUCTION

Games are structured form of play, usually undertaken for entertainment or fun, and sometimes used as an educational tool. They can be played alone, in teams, or online, by amateurs or by professionals.

Key components of games are goals, rules, challenge, and interaction. Games generally involve mental or physical stimulation, and often both. Many games help develop practical skills, serve as a form of exercise, or otherwise perform an educational, simulation, or psychological role.

Missile Command is one such online game which has set rules, a format that is played by a single player. The game prevents boredom, improves concentration levels, improves speed of reactivity to the actions and improves control over actions of the players.

Missile Command is an arcade game and can only be played by 1 player. In the game there are many meteors fall from sky which may hit the city, so that, city may get destroyed. The main aim of the player is to launch the counter missiles against meteors and save the city.

The game screen contains a city, and it has 3 missile launchers these missile launchers are capable of launching missiles. In the game each missile launcher is provided with only fixed number of missiles, in our game the number is 10. If the 10 missiles are used by the player, then missile launcher becomes inactive. Missile launchers are invulnerable from destruction. If the missile launcher has more than 4 missiles in it, then it is shown in bold, else it is shown with only outline of it

So, here in the game when player presses on the game screen, a cross hair is displayed (which is the destination of the missile), then missile is launched from nearest possible missile launcher. When missile hits the meteor, explosion occurs.

Game Components:

✓ The game screen contains a city that has 3 missile launchers, and 6 oval shaped objects that represent city.

Missile launchers don't affect by meteors, but city can be destroyed from meteors hit.

Game play:

Missile Command is an arcade game where it tests the speed of reactivity of the player. Here, in the game, many meteors fall from sky which may hit the city and that causes city to destroy. So, the player has to save the city from getting destroyed.

The player has to explode the meteor by launching a missile from missile launcher. Missile is launched from the missile launcher when he presses on the game screen. Then a cross hair is displayed on the screen which depicts the missile's destination position and that is place where missile is targeted. Missile is launched to the position where cross hair appears i.e., where player clicks the screen.
When missile hits the meteor explosion occurs, this process occurs repetitively until number of missiles is equal to 0, then missile hit the city later city ruins and game ends.

End of The Game:

When the total number of missiles available in the missile launcher becomes zero, the there is no chance to launch the missile. So, missile launchers become inactive. Then meteors directly hit the city which causes city to destroy. Then game ends.

2.SYSTEM ANALYSIS

2.1 Existing System

Missile Command is a 1980 arcade game developed and published by Atari, Inc. It was designed by Dave Theurer.

The game is played by moving a crosshair across the sky background via a trackball and pressing one of three buttons to launch a counter-missile from the appropriate battery. Counter-missiles explode upon reaching the crosshair, leaving a fireball that persists for several seconds and destroys any enemy missiles that enter it. There are three batteries, each with ten missiles; a missile battery becomes useless when all its missiles are fired, or if the battery is destroyed by enemy fire. The missiles of the central battery fly to their targets at much greater speed; only these missiles can effectively kill a smart bomb at a distance.

The game is staged as a series of levels of increasing difficulty; each level contains a set number of incoming enemy weapons. The weapons attack the six cities, as well as the missile batteries; being struck by an enemy weapon results in the destruction of the city or missile battery.

2.2 Proposed System

The project aims to develop a game called **Missile Command.**

Missile Command is an arcade game and can only be played by a single player. In the game there are many meteors falling from sky which may hit the city, so that, city may get destroyed. So, here the aim of the player is to shoot the meteors using missiles and save the city from destruction.

The game screen contains a city, and it has 3 missile launchers these missile launchers are capable of launching missiles. In the game each missile launcher is provided with only fixed number of missiles, in our game the number is 10. If the 10 missiles are used by the player, then missile launcher becomes inactive. Missile launchers are invulnerable from destruction. If the missile launcher has more than 4 missiles in it, then it is shown in bold, else it is shown with only outline of it.

In the game when player presses on the game screen, a cross hair is displayed (which is the destination of the missile), then missile is launched from nearest possible missile launcher. When missile hits the meteor, explosion occurs.

Here, the main aim of the player is shoot missiles using mouse control provided and save the city from destruction.

2.3 Feasibility Study

An important outcome of preliminary investigation is the determination that the system request is feasible. This is possible only if it is feasible within limited resource and time.

The different feasibilities that must be analyzed are: -

- Operational Feasibility
- Economic Feasibility
- Technical Feasibility

Operational Feasibility:

Operational feasibility deals with the study of prospects of the system to be developed. This system is a game named Missile Command. Based on the study, the system is proved to be operationally feasible.

Economic Feasibility:

Economic feasibility or Cost-benefit is an assessment of the economic justification for the computer-based project. As hardware and software was installed from the beginning and for lots of purposes thus the cost on hardware and software is low. So, the project is economically feasible.

Technical Feasibility:

Technical Feasibility is the process of validating the technology assumptions, architecture and design of a product or project. As we need import only few packages like java. awt package, java. awt. event package, java. applet. Applet class, java. util. Vector class for creation of GUI components, creating graphics and to associate event handling for them and threading concepts which are inbuilt packages from jdk the proposed system is technically feasible.

3.SYSTEM REQUIREMENTS SPECIFICATION

3.1 Software Requirements

- **Programming Language:** Java 1.8 • **Operating System:** Windows 10

3.2 Hardware Requirements

- **Processor:** I3 • **RAM:** 4GB • **Hard Disk:** 1TB

3.3 Functional Requirements

In Software Engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, behavior, outputs. Functional requirements may be calculations, technical details, data manipulation and supposed to accomplish. The plan for implementing functional requirements is detailed in the system design. In requirements engineering, functional requirement specify particular results of a system. Functional requirements drive the application architecture of a system. The following are the functional requirements of our system: -

- System should be able to process the position of the mouse control.

4.SYSTEM DESIGN

4.1 Introduction

System design is the process of designing the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system.

System Analysis is the process that decomposes a system into its component pieces for the purpose of defining how well those components interact to accomplish the set requirements. The purpose of the System Design process is to provide sufficient detailed data and information about the system and its system elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture.

The purpose of the design phase is to plan a solution of the problem specified by the requirement document. This phase is the first step in moving from problem domain to the solution domain. The design of a system is perhaps the most critical factor affecting the quality of the software, and has a major impact on the later phases, particularly testing and maintenance.

The output of this phase is the design document. This document is similar to a blueprint or plan for the solution, and is used later during implementation, testing and maintenance. The design activity is often divided into two separate phase-system design and detailed design. System design, which is sometimes also called top-level design, aims to identify the modules that should be in the system, the specifications of these modules, and how they interact with each other to produce the desired results. At the end of system design all the major data structures, file formats, output formats, as well as the major modules in the system and their specifications are decided.

A design methodology is a systematic approach to creating a design by application of set of techniques and guidelines. Most methodologies focus on system design. The two basic principles used in any design methodology are problem partitioning and abstraction. A large system cannot be handled as a whole, and so for design it's partitioned into smaller systems. Abstraction is a concept related to problem partitioning. When partitioning is used during design, the design activity focuses on one part of the system at a time. Since the part being designed interacts with other parts of the system, a clear understanding of the interaction is essential for property designing the part.

The system, we wish to design is a game named 'Missile Command', that interacts with user with the help of mouse inputs. This system can be implemented using Java programming language with the help of some inbuilt packages, and its methods available.

The brief explanation of the system to be designed is, there are many meteors falling from sky which may hit the city, so that, city may get destroyed. So, the player must shoot the meteors using missiles and save the city from destruction.

The game screen contains a city, and it has 3 missile launchers these missile launchers can launch missiles. In the game each missile launcher is provided with only fixed number of missiles, in our game the number is 10. If the 10 missiles are used by the player, then missile launcher becomes inactive. Missile launchers are invulnerable from destruction. If the missile launcher has more than 4 missiles in it, then it is shown in bold, else it is shown with only outline of it.

In the game when player presses on the game screen, a cross hair is displayed (which is the destination of the missile), then missile is launched from nearest possible missile launcher. When missile hits the meteor, explosion occurs.

The system design is: -

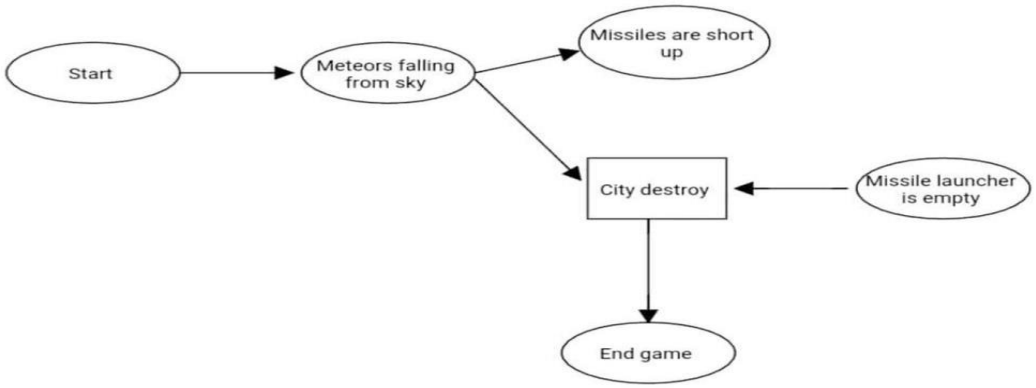

Figure 4.1.1 system design

4.2. UML Diagrams

UML (Unified Modeling Language) Diagrams is a rich visualizing model for representing the system architecture and design. These diagrams help us to know the flow of the system.
Some of them are: -

- Use case diagram
- Class diagram
- Sequence diagram
- Activity diagram
- Collaboration diagram

4.2.1 Usecase Diagram:

A **Use Case Diagram** in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose

is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. Interaction among actors is not shown on the use case diagram. If this interaction is essential to a coherent description of the desired behavior, perhaps the system or use case boundaries should be re-examined.

Use cases:

A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

Actors:

An actor is a person, organization, or external system that plays a role in one or more interactions with the system.

System boundary boxes:

A rectangle is drawn around the use cases, called the system boundary box, to indicate the scope of system.

Include:

In one form of interaction, a given use case may include another. "Include is a Directed Relationship between two use cases, implying that the behavior of the included use case is inserted into the behavior of the including use case.

The first use case often depends on the outcome of the included use case. This is useful for extracting truly common behaviors from multiple use cases into a single description. The notation is a dashed arrow from the including to the included use case, with the label "«include»".

Extend:

In another form of interaction, a given use case may extend another. This relationship indicates that the behavior of the extension use case may be inserted in the extended use case under some conditions. The notation is a dashed arrow from the extension to the extended use case, with the label "«extend»". Modelers use the «extend» relationship to indicate use cases that are "optional" to the base use case.

Generalization:

In the third form of relationship among use cases, a generalization / specialization relationship exists. A given use case may have common behaviors, requirements, constraints, and assumptions with a more general use case. In this case, describe them once, and deal with it in the same way, describing any differences in the specialized cases.

Identified Use Cases

The "user model view" encompasses a problem and solution from the preservative of those individuals whose problem the solution addresses. The view presents the goals and objectives of the problem owners and their

requirements of the solution. This view is composed of "use case diagrams". These diagrams describe the functionality provided by a system to external integrators. These diagrams contain actors, use cases, and their relationships. overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. Interaction among actors is not shown on the use case diagram. If this interaction is essential to a coherent description of the desired behavior, perhaps the system or use case boundaries should be re-examined. Alternatively, interaction among actors can be part of the assumptions used in the use case.

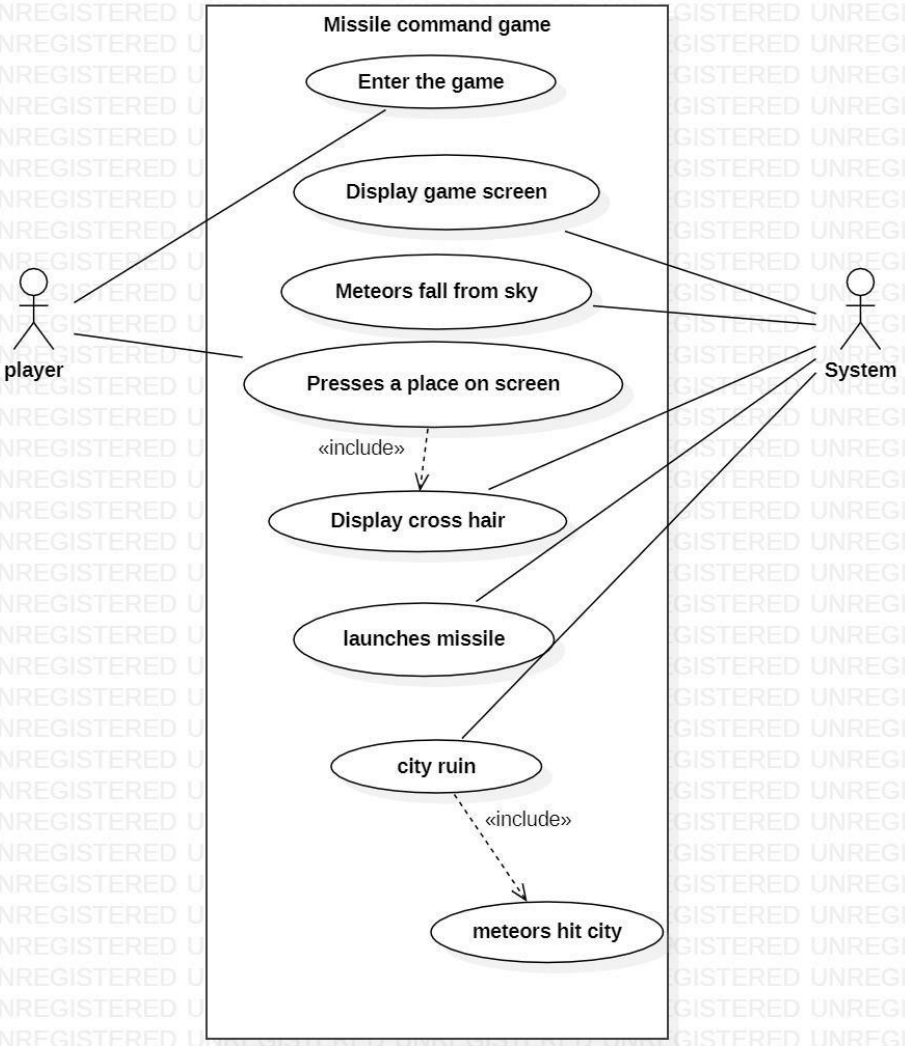

Fig 4.2.1 Usecase Diagram

1.Actors: - player, system

2.Usecases: - enter the game, display game screen, meteors fall from sky, presses a place on screen, display crosshair, launches missile, meteors hit city, city ruin

4.2.2 Class Diagram:

A **class diagram** in UML is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. The class diagram is the main building block of objectoriented modelling. It is used for general conceptual modelling of the structure of the application, and for detailed modelling translating the models into programming code. The classes in a class diagram represent both the main elements, interactions in the application, and the classes.

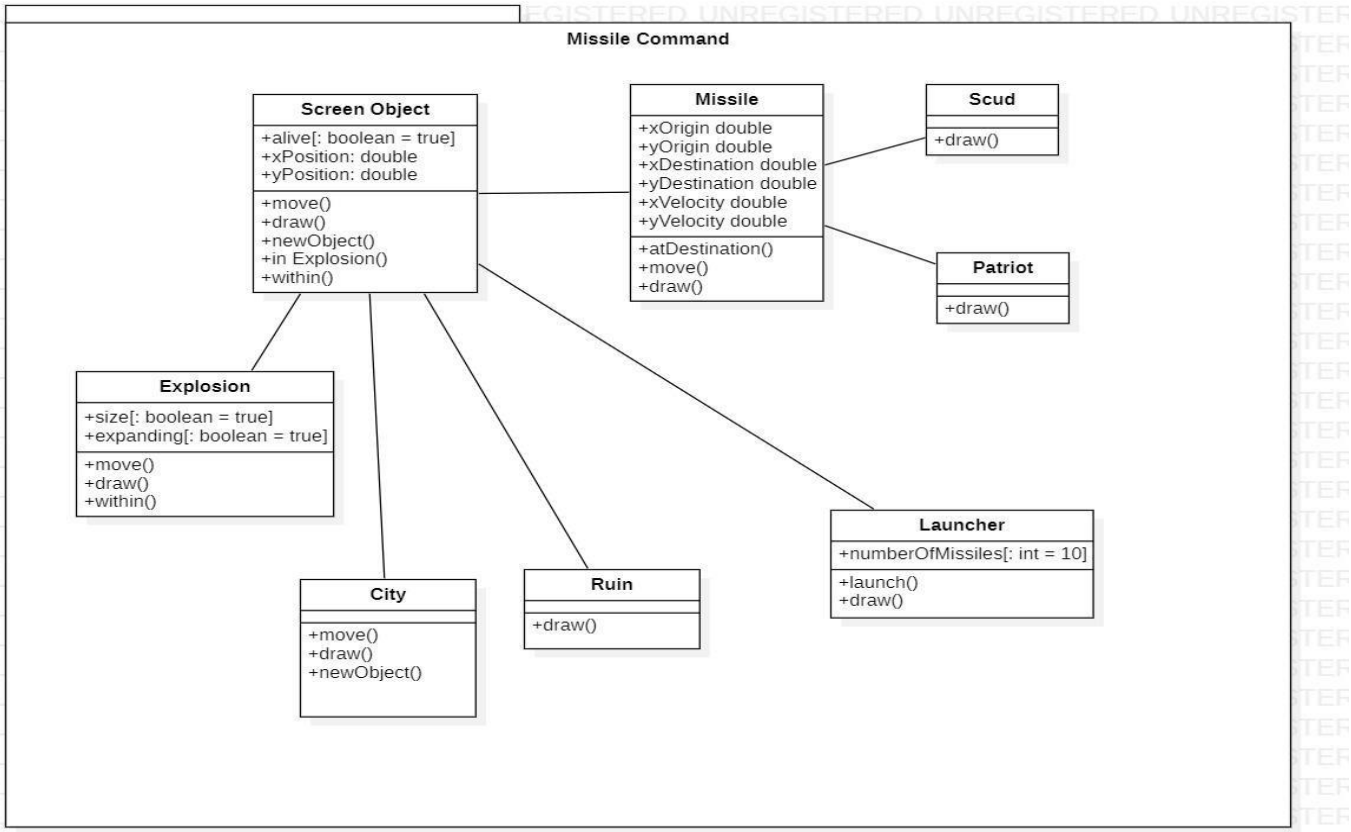

Fig 4.2.2 class diagram

Classes: - ScreenObject, Missile, Scud, Patriot, Explosion, City, Ruin, Launcher

4.2.3 Sequence Diagram:

A **Sequence diagram** is an interaction diagram that emphasizes the time ordering of messages. It is consisting of a set of objects and their relationships, including the messages that maybe dispatched among them. Sequence diagrams address the dynamic view of system. Sequence diagram is a two-dimension nature.

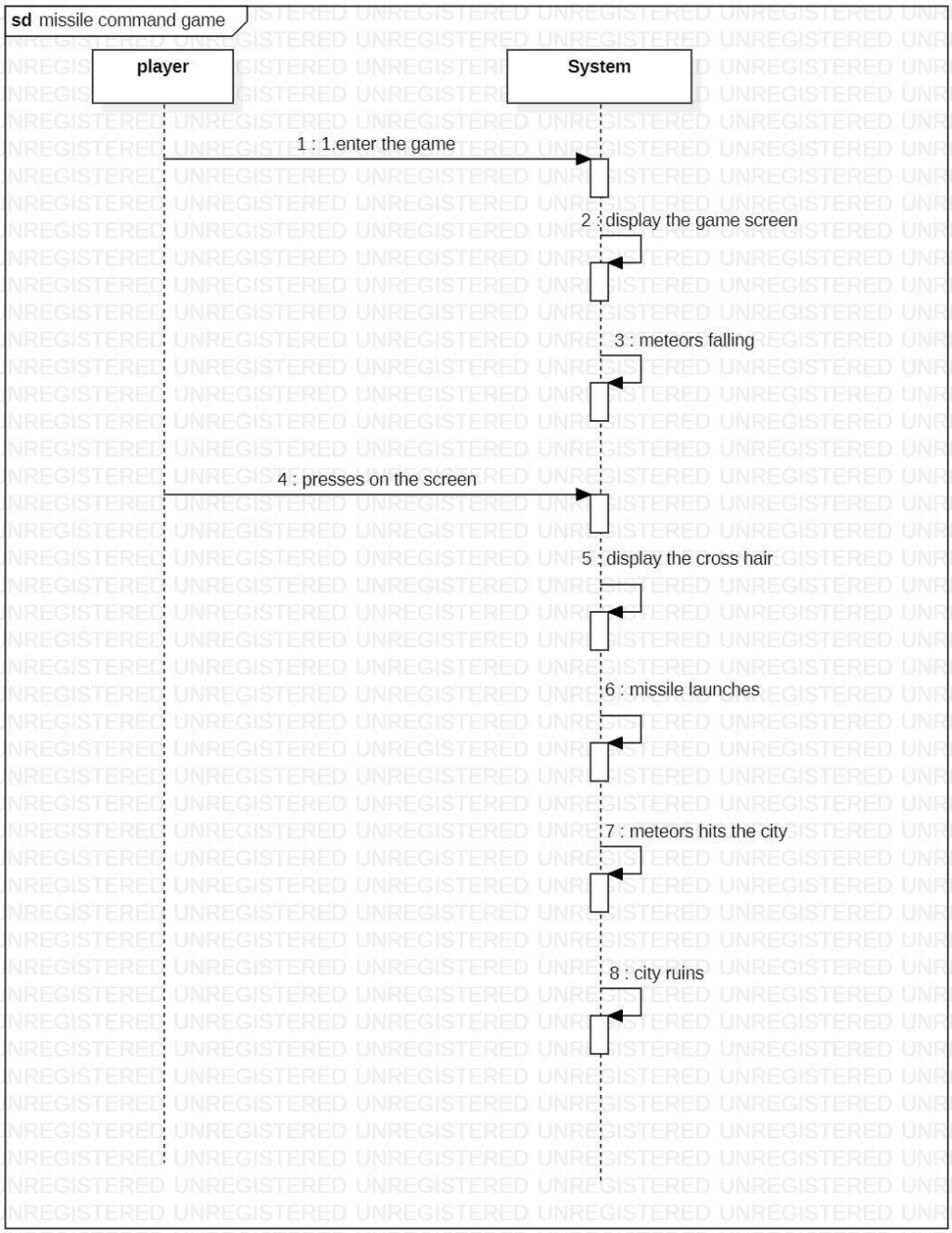

Fig 4.2.3 sequence diagram

1. LIFELINE: Player, System

2. Sequence flow occurs according to their order.

4.2.4 Activity Diagram:

An **activity diagram** is a special kind if state chart diagram that shows the flow from activity to activity within a system. Activity Diagrams address the dynamic view of a system.

They are especially important in modeling the function of a System and emphasize the flow of control among objects. It describes the flow of activity in system. It shows connection between one activity of System to another activity. Before drawing activity diagram, you should list activity of user and connection between both activities.

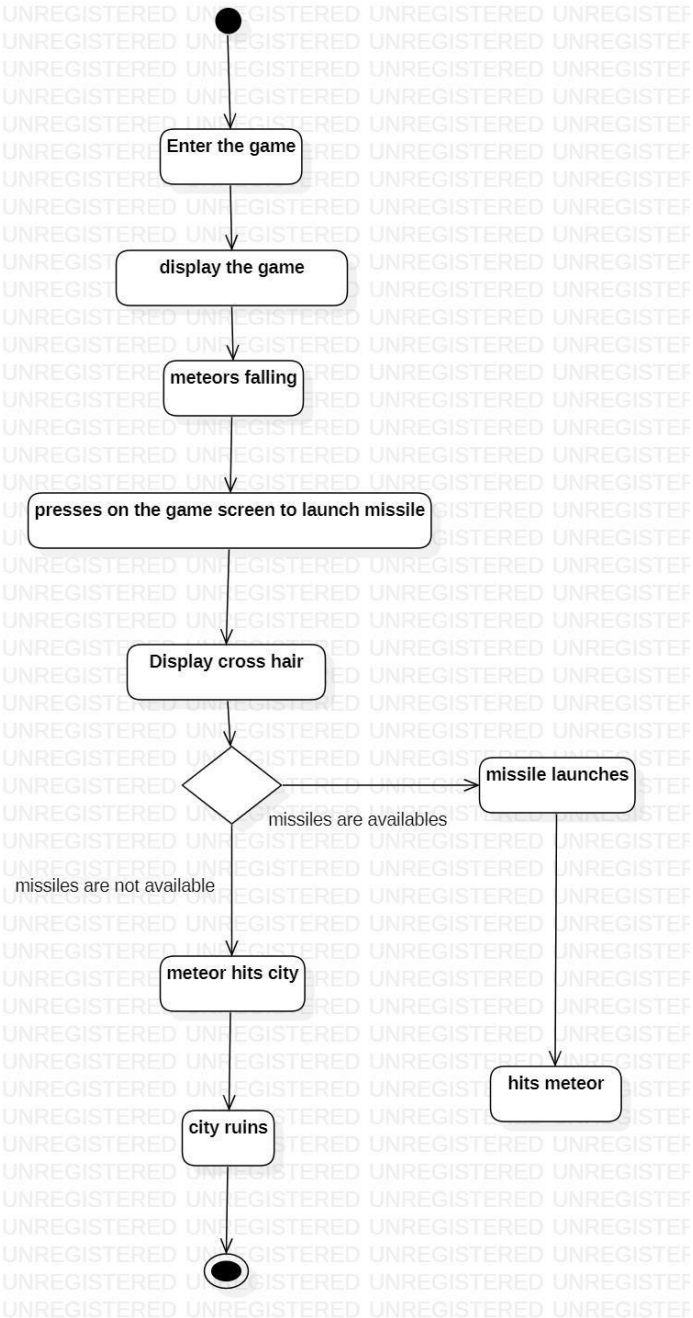

Fig 4.2.4 activity diagram

1. Initially, player enters the game.

2. Gaming screen is displayed by the system.
3. Screen contains meteors falling movement.

4. When player presses on screen crosshair appears.

5. If missiles are available missile is launched by missile launcher and it hits meteor else meteor hits city and city ruins.

4.2.5 COLLOBORATION DIAGRAM:

A **collaboration diagram** is an interaction diagram that emphasizes the structural organization of the objects that send receive message. A collaboration diagram is very similar to sequence diagram. Collaboration diagram shows the objects and their association with other objects. Modelling objects in a system and representing the association between the objects as links are easily represented in a collaboration diagram. Sequence diagrams and collaboration diagrams shows same information but sequence diagram focus on the temporal aspect and collaboration diagram focus on communication between the objects of system.

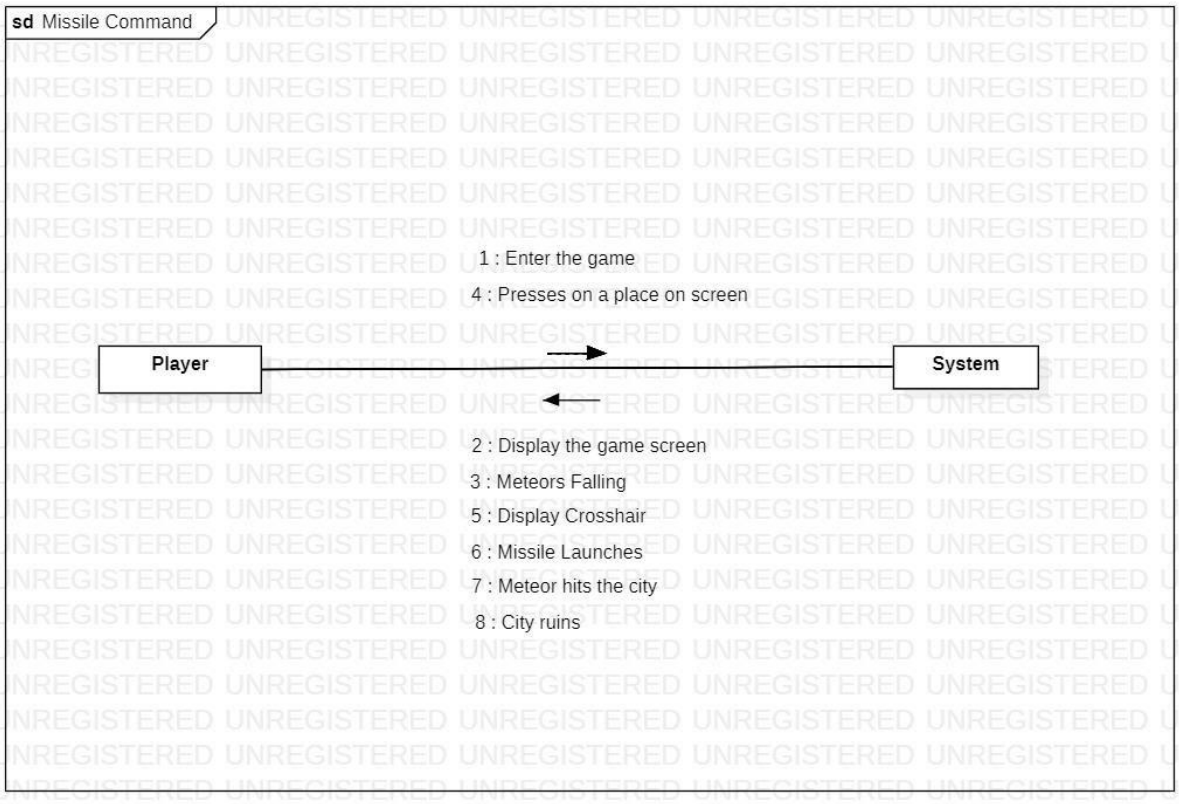

Fig 4.2.5 collaboration diagram

Objects: Player, System

5. SYSTEM IMPLEMENTATION

5.1 Introduction

The purpose of system implementation can be summarized as follow making the new system available to the prepared set of users (the deployment) and positioning ongoing support and maintenance of the system within the performing organization (the transaction). At a finer necessary to educate the consumer on the use of system, placing the newly developed system into production, confirming that business functions that interact with the system and functioning properly.

Transitioning the system support responsibilities involve changing from a system development to the system and maintenance mode of operation, with ownership of the new system moving from the project team to the performing organization.

A key difference between system implementation and all other phases of lifecycle is that all project activities up to this point have been performed in safe, protected and check your environments. It is through the careful planning, execution, and management of system implementation activities that the project team can minimize the likelihood of these occurrences and determine appropriate contingency plans in the event of the problem.

The system to be implemented is Missile Command game which is an arcade game and can only be played by single player. In the game there are many meteors fall from sky which may hit the city, so that, city may get destroyed.

The city is provided with 3 missile launchers which consists of 10 missiles each, these missile launchers are invulnerable from destruction and become inactive only when the number of missiles left with is zero. So, here the main aim of the player is to shoot the meteors by missiles using mouse control and save the city from destruction.

In the game when player presses on the game screen, a cross hair is displayed (which is the destination of the missile), then missile is launched from nearest possible missile launcher. When missile hits the meteor, explosion occurs. Here, in the game, the main aim of the player is shoot missiles using mouse control provided and save the city from destruction.

5.2 Project Modules

- Screen Objects
- Missile
- City
- Missile Launcher
- Explosion

Screen Objects

Screen Objects is a module which creates all the instances that appear on the gaming screen. The instances include missile launchers, missiles, city, explosion. All the instances remain on screen until they are exploded. (e.g., city explodes by meteor collision until then city remains in the game screen).

Screen Objects class written in the code is the super class of all the classes that create objects on game screen. The user defined methods like move, in Explosion, within are implemented in this class, and any class which does that action overrides the methods here according to their usage.

Missile

Missiles are Screen Objects move about on the screen. Missiles are launched from missile launcher to hit the meteor. They are timed to reach the target (meteor) then meteor disappears. These missiles are shot up only when player presses on any of position in gaming screen. There are only limited number of missiles in each missile launcher.

Missiles are shot up against meteor only when player presses on the screen by missile launcher. There are only limited number of missiles available in each missile launcher i.e; 10 missiles are provided for each missile launcher, later it becomes inactive.

Event handling is associated in order to launch missile from missile launcher when player presses. For this mouse listener interface and mouse motion listener interface are implemented for this. Since, an interface is implemented all the methods associated with it must be overridden, here, the methods used are mouse Moved and mouse Pressed so only they are implemented with required code and other all are implemented empty.

Here, in the picture, pink colored lines represent missiles.

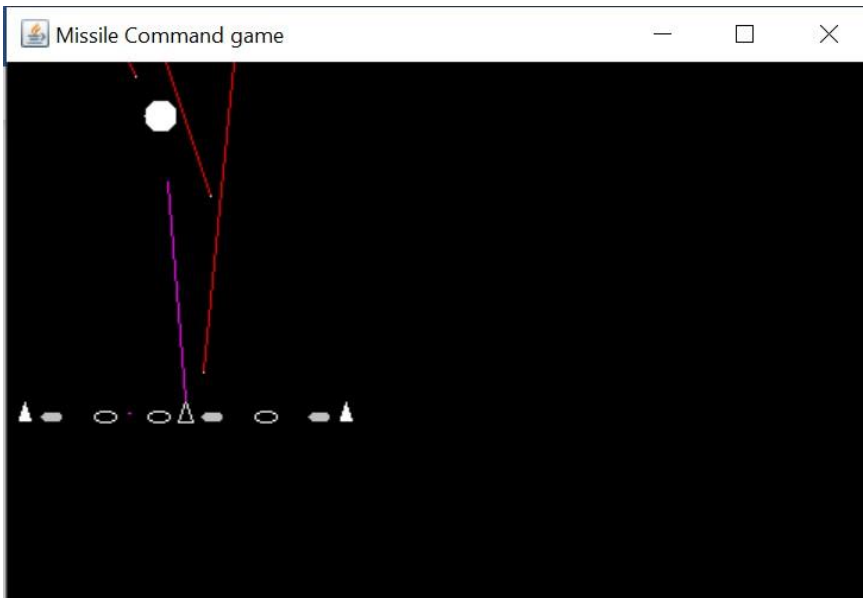

Fig 5.2.1 Missile (Pink coloured)

City

City is the land it contains some missile launchers. Cities are vulnerable to explosions. These are designed using packages like java. applet. Applet class and java. awt package. Some of the inbuilt methods used are abstract void fill Oval (int x, int y, int width, int height), abstract void draw Oval (int x, int y, int width, int height), abstract void set Color (Color c).

Here, in the picture there 6 oval shaped objects, these represent city.

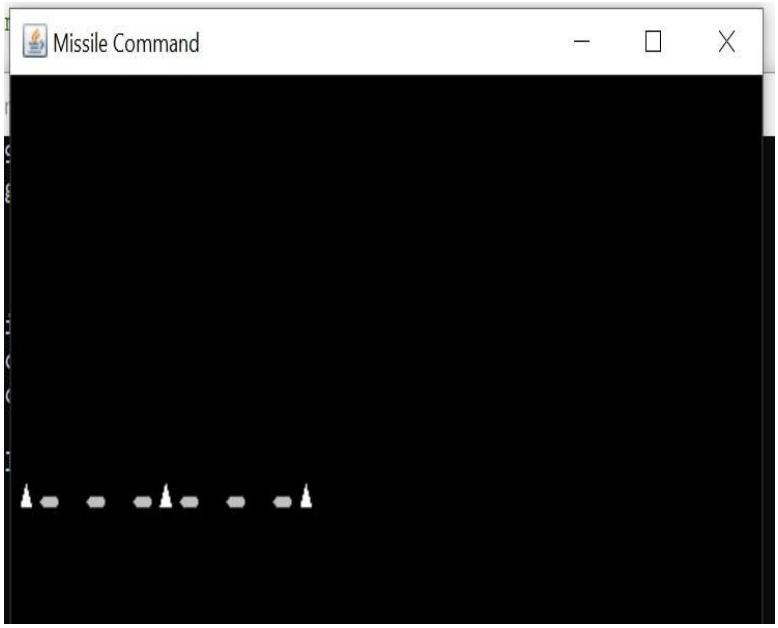

Fig 5.2.2 City

Missile Launcher

Missile Launcher launches the missile. The city is provided with 3 missile launchers which consists of 10 missiles each, these missile launchers are invulnerable from destruction and become inactive only when the number of missiles left with is zero.

These are designed using packages like java. applet. Applet class and java. awt package. Some of the inbuilt methods used are public abstract void draw Polygon(int[] x, int [] y, int n points)**,** public abstract void fill Polygon(int[] x Points, int[] y Points, int n points), abstract void set Color (Color c).

Here, In the picture there 3 polygons those represent missile launchers. When the number of missiles in a missile launcher is more than 4 then it is visualized in bold else, only outline is visualized on the screen and when missile launcher becomes inactive it disappears from screen.

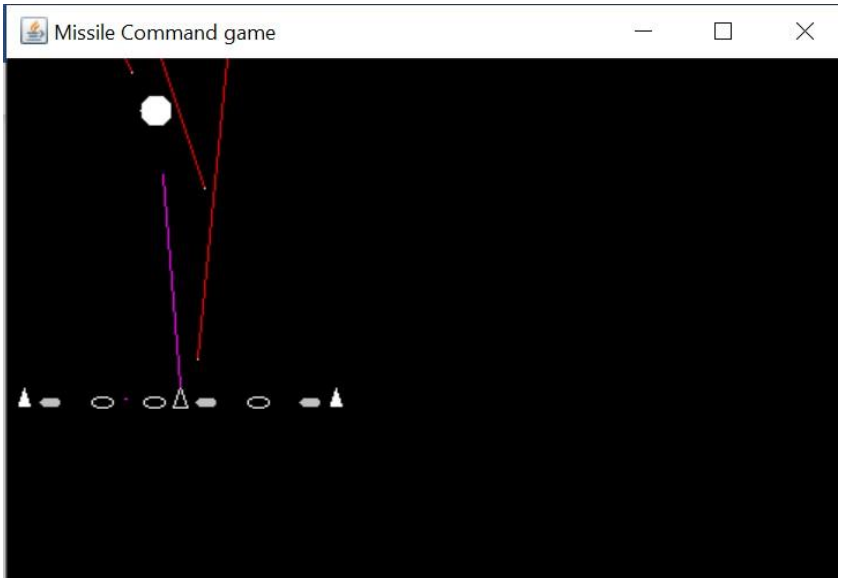

Fig 5.2.3 Missile Launcher (Polygon shaped)

Explosion

Explosions are produced when missile hits the meteor or when meteor hits the missile launcher. Explosions are Screen Objects that expand and shrink i.e., to make it visualize like smoke after explosion.

These are designed using packages like java. applet. Applet class and java. awt package. Some of the inbuilt methods used are abstract void fill Oval (int x, int y, int width, int height), void set XOR Mode (Color c), abstract void set Color (Color c)

Here, in the picture the explosion is seen where missile hits missile launcher in different colors like magenta, orange and white.

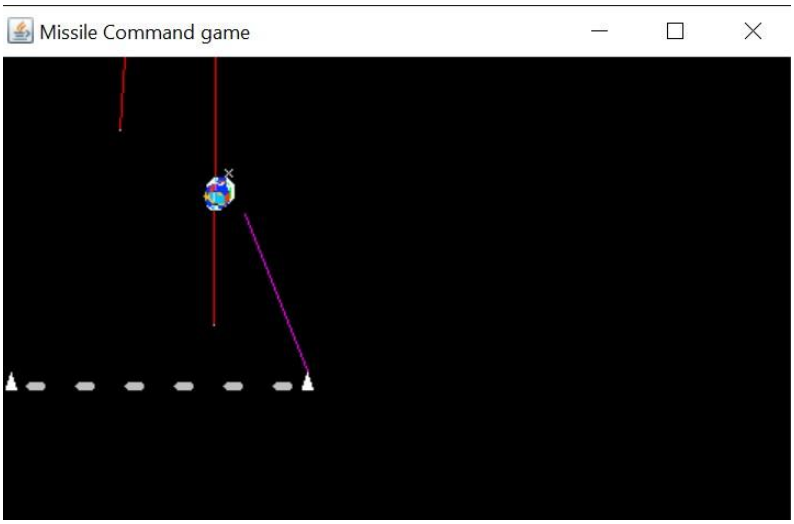

Fig 5.2.4 Explosion

5.3 Screens

Screen 5.3.1: This initial screen has city, missile launchers. In the screen city is in oval shaped and missile launcher is in cone shaped.

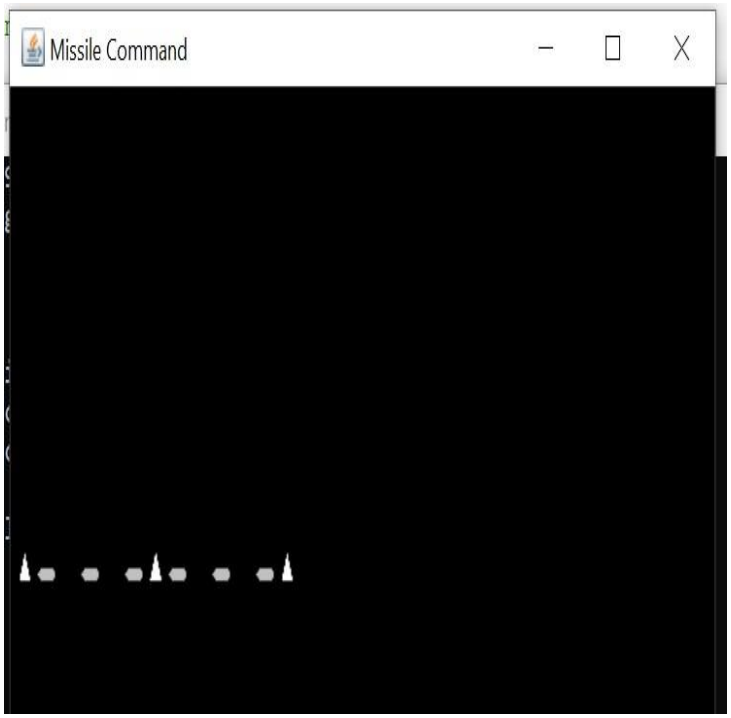

Figure 5.3.1 initial screen

Screen 5.3.2: This picture displaying falling of meteors.

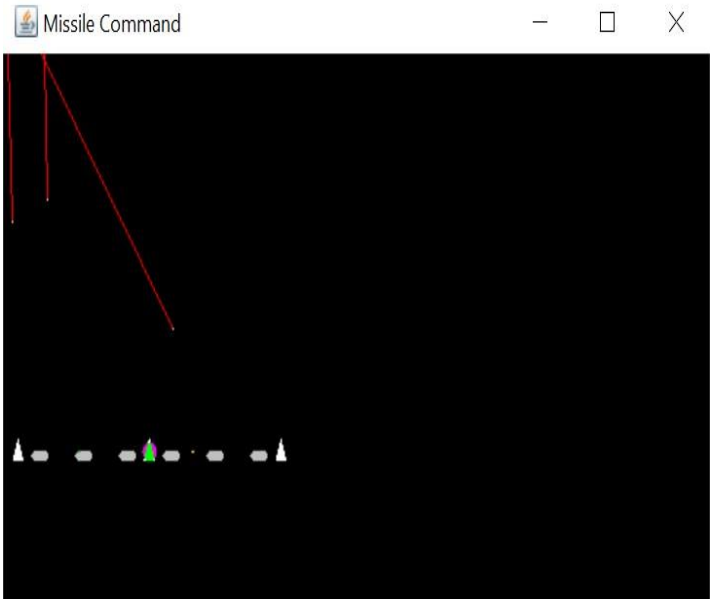

Figure 5.3.2 Picture displaying falling of meteors.

Screen 5.3.3: This picture displays missile launching (pink colored in the picture). This action happens only when player presses on the screen.

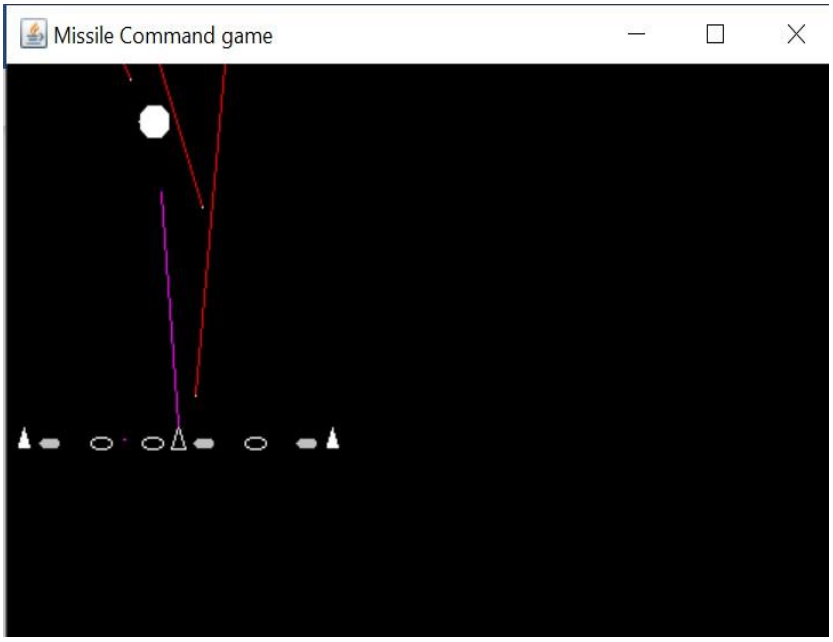

Fig 5.3.3 Missile Launching

Screen 5.3.4: This picture displaying explosion. Explosion is seen in magenta, orange and white colors. Explosion occurs when missile hits the meteor.

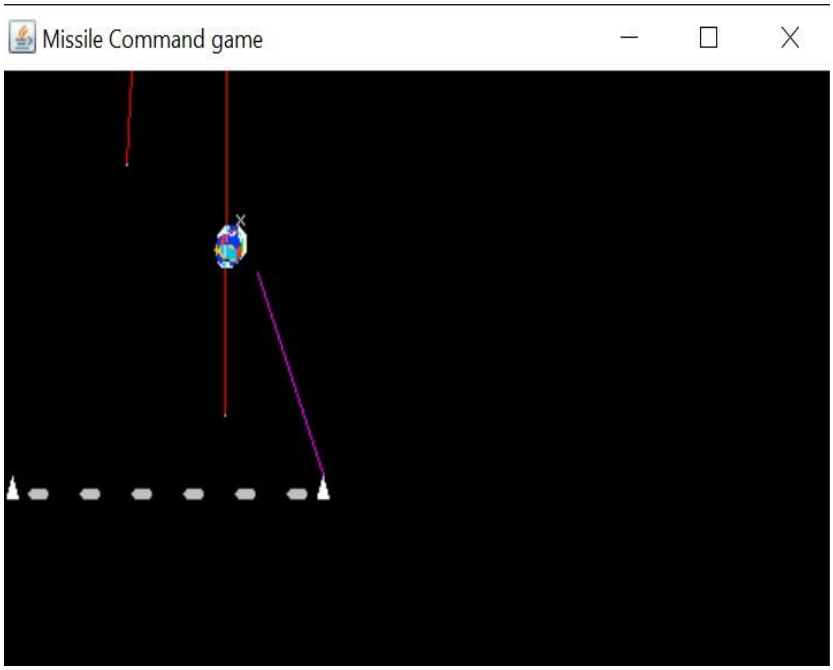

Figure 5.3.4 Picture displaying explosion.

Screen 5.3.5: This picture displays the screen when all missiles are completed, so there is no chance to save the city then city gets ruined. At first city is seen in bold display after getting ruined it is seen with only it's outline. It displays the end of the game.

Fig 5.3.5 City ruined

6.SYSTEM TESTING

6.1 Introduction

Software Testing is an important element of the software quality assurance and represents the ultimate review of specification, design, and coding. The increasing feasibility of software as a system and the cost associated with the software failures are motivated forces for III planned through testing.

System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements.

In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behavior of a component or a system when it is tested.

System Testing is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or in the context of both. System testing tests the design and behavior of the system and the expectations of the customer. It is performed to test the system beyond the bounds mentioned in the software requirements specification (SRS).

System Testing is basically performed by a testing team that is independent of the development team that helps to test the quality of the system impartial. It has both functional and non-functional testing.

System Testing is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system. Ultimately, the software is interfaced with other software/hardware systems. System Testing is a series of different tests whose sole purpose is to exercise the full computer-based system.

System Testing is done after Integration Testing. This plays an important role in delivering a high-quality product.
Verification: Confirmation by examination and provisions of objective evidence that specified requirements have been fulfilled. If an application has three modules A, B, and C, then testing done by combining the modules A & B or module B & C or module A & C is known as Integration testing. Integrating all the three modules and testing it as a complete system is termed as System testing.

TESTING OBJECTIVES

These are several rules that can save as testing objectives:

- Testing is a process of executing program with the intent of finding an error.
- A good testcase is one that has a high probability of finding an undiscovered error.

Test Levels

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or darkness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.2. Testing Methods

6.2.1 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure, and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

6.2.2 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document.
By black box testing, it is found that outputs got matches the different inputs given.

6.2.3 Unit Testing

Unit testing is the first level of testing and is often performed by the developers themselves. It is the process of ensuring individual components of a piece of software at the code level are functional and work as they were designed to. Developers in a testdriven environment will typically write and run the tests prior to the software or feature being passed over to the test team. Unit testing can be conducted manually but automating the process will speed up delivery cycles and expand test coverage. Unit testing will also make debugging easier because finding issues earlier means they take less time to fix than if they were discovered later in the testing process. Test Left is a tool that allows advanced testers and developers to shift left with the fastest test automation tool embedded in any IDE.

6.2.4 Integration Testing

Integration tests are designed to test integrated software components to determine if they run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

After each unit is thoroughly tested, it is integrated with other units to create modules or components that are designed to perform specific tasks or activities. These are then tested as group through integration testing to ensure whole segments of an application behave as expected (i.e, the interactions between units are seamless). These tests are often framed by user scenarios, such as logging into an application or opening files. Integrated tests can be conducted by either developers or independent testers and are usually comprised of a combination of automated functional and manual tests. By integration testing, it is seen that

all the modules (in the game) that are designed separately are integrating in a manner such that user gets the required output.

6.2.5 Validation Testing

Validation Testing ensures that the product meets the client's needs. It can also be defined as to demonstrate that the product fulfills its intended use when deployed on appropriate environment. By validation testing, it is seen that the project meets its needs.

6.2.6 System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test.

System testing is a black box testing method used to evaluate the completed and integrated system to ensure it meets specified requirements. The functionality of the software is tested from end-to-end and is typically conducted by a separate testing team than the development team before the product is pushed into production.

6.3 Testcases

Testcase 6.3.1: User Interface
Check for position and movement of objects.

Testcase 6.3.2: Performance Check
for flow of game.

Testcase 6.3.3 Pause
Game pauses when pressed on minimize button and resumes when maximized.

Testing the display of gaming screen for some actions
Testcase 6.3.4: - When all the missiles are completed, then missile launcher display is disappeared. Later city is ruined because there are no missiles available. It represents game end.

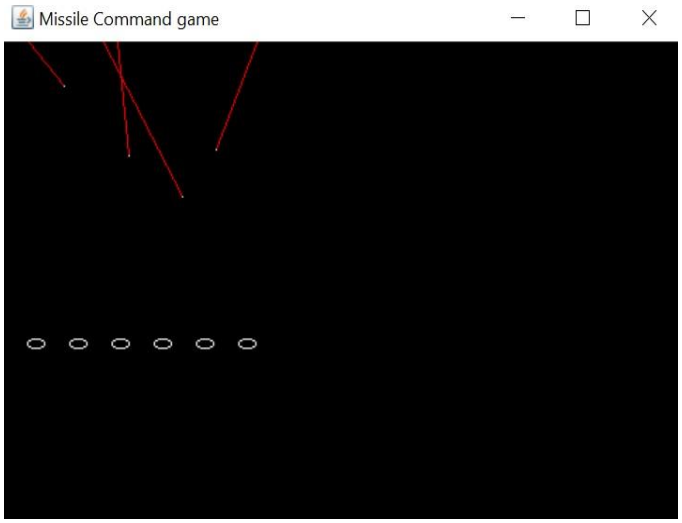

Fig 6.3.1 All the missiles are used, and city is ruined.

Testcase 6.3.5: - When missiles are not shot against the meteor. Then city is ruined, and all the missiles provided are available with the missile launcher.

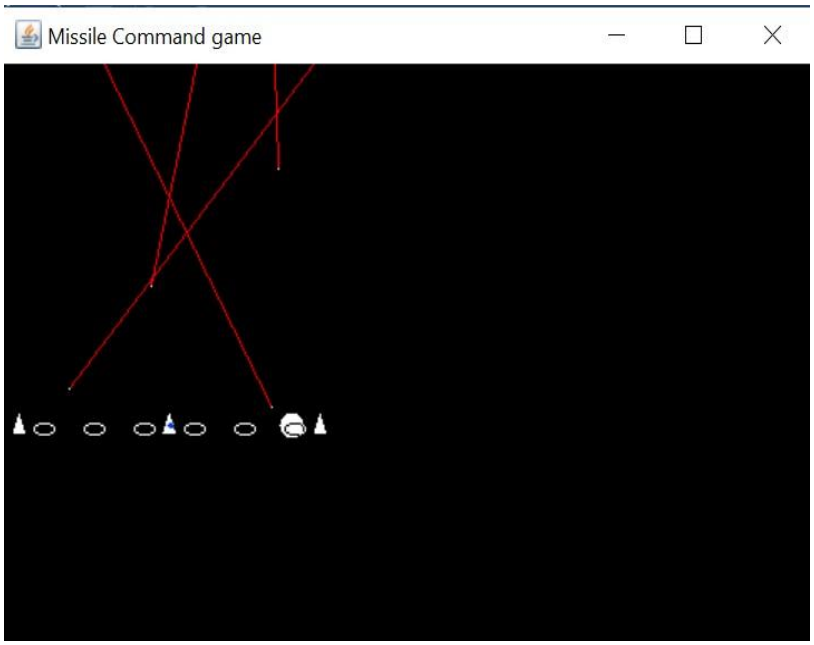

Fig 6.3.1 All the missiles are available, and city is ruined.

Testcase 6.3.6: - When the number of missiles in the missile launcher are less than 4, the display of missile launcher is converted from bold to having only outline.

Here, in the fig 6.3.3, 1st,3rd missile launchers have number of missiles more than 4 and the 2nd missile launcher has less than 4 missiles in it.

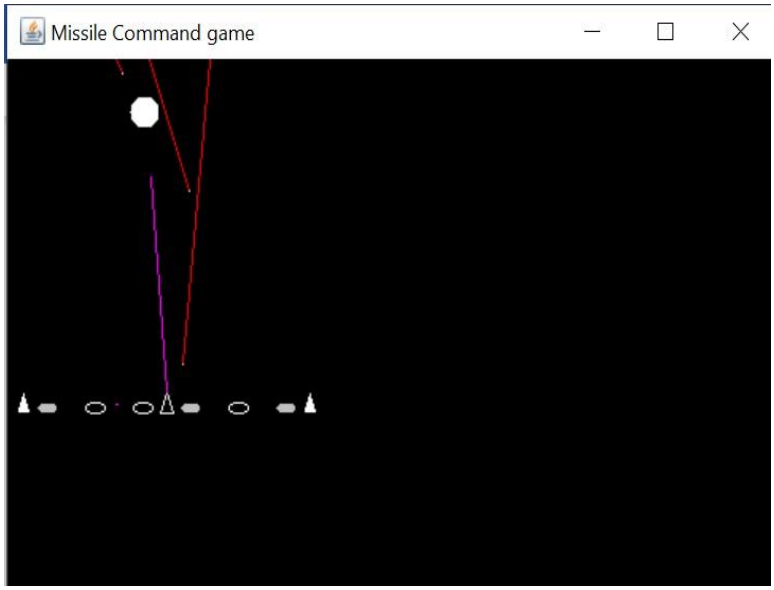

Fig 6.3.3 Displays the difference in visibility when the number of missiles in a missile launcher are more than 4 and when they are less than 4.

7. CONCLUSION

Now a days people are interested to play online games. So, we decided to design an online game. Missile Command is a very interesting online game, and it is easy to play this game. This game improves the control over actions of the players.

Missile Command is an arcade game and can only be played by a single player. In the game there are many meteors falling from sky which may hit the city, so that, city may get destroyed. So, here the aim of the player is to shoot the meteors using missiles and save the city from destruction.

The game screen contains a city, and it has 3 missile launchers these missile launchers are capable of launching missiles. In the game each missile launcher is provided with only fixed number of missiles, in our game the number is 10. If the 10 missiles are used by the player, then missile launcher becomes inactive. Missile launchers are invulnerable from destruction. If the missile launcher has more than 4 missiles in it, then it is shown in bold, else it is shown with only outline of it.

In the game when player presses on the game screen, a cross hair is displayed (which is the destination of the missile), then missile is launched from nearest possible missile launcher. When missile hits the meteor, explosion occurs.

Here, the main aim of the player is shoot missiles using mouse control provided and save the city from destruction.

Later we will implement the game with associating database to it which helps in storing players scores and display high score.

8.BIBLIOGRAPHY

- https://en.wikipedia.org/wiki/Missile_Command
- https://youtu.be/nokIGklnBGY
- https://docs.oracle.com/javase/7/docs/api/java/awt/package-use.html
- https://docs.oracle.com/javase/7/docs/api/java/awt/Graphics.html
- https://www.javatpoint.com/event-handling-in-java
- https://www.tutorialspoint.com/awt/awt_mouse_event.htm
- https://www.javatpoint.com/multithreading-in-java
- https://docs.oracle.com/javase/7/docs/api/java/util/Vector.html

9.APPENDIX

9.1 Introduction to Java

Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general-purpose programming language intended to let application developers write once, run anywhere (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture. The syntax of Java is similar to C and C++ but has fewer low-level facilities than either of them. The Java runtime provides dynamic capabilities (such as reflection and runtime code modification) that are typically not available in traditional compiled languages. As of 2019, Java was one of the most popular programming languages in use according to GitHub, particularly for client-server web applications, with a reported 9 million developers.

Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle) and released in 1995 as a core component of Sun Microsystems' Java platform. The original and reference implementation Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licenses. As of May 2007, in compliance with the specifications of the Java Community Process, Sun had relicensed most of its Java technologies under the GNU General Public License. Oracle offers its own Hotspot Java Virtual Machine; however the official reference implementation is the OpenJDK JVM which is free open source software and used by most developers and is the default JVM for almost all Linux distributions.

Java Buzzwords

- **Simple:** Java inherits all the best features from the programming languages like C, C++ and thus makes it really easy for any developer to learn with little programming experience. The concept of Object-Oriented programming was not invented by Java, but it was just adopted by the Java team. Programming notation of Java is not different from the programming language like C and C++ which makes developers have little trouble to learn Java.

- **Secure:** When Java programs are executed, they don't instruct commands to the machine directly. Instead, Java Virtual machine reads the program (Bytecode) and convert it into the machine instructions. This way any program tries to get illegal access to the system will not be allowed by the JVM. Allowing Java programs to be executed by the JVM makes Java program fully secured under the control of the JVM.

- **Portable:** Java programs are portable because of its ability to run the program on any platform and no dependency on the underlying hardware / operating system.

- **Object Oriented Programming Language:** Java is object-oriented programming language but everything in Java are not objects. Java manages to maintain balance and adopted what make sense in the current situation. The object-oriented model in Java is simple and easy to extend and also the primitive types such as integers, are retained for high-performance.

- **Robust:** Following features of Java make it Robust.
    - Platform Independent
    - Object Oriented Programming Language ○ Memory management ○ Exception Handling

- **Platform Independent:** Java program are written once and executed on any platform this makes the job of developer easier to develop programs and not code machine dependent coding.

- **Multithreaded:** Java allows you to develop program that can do multiple task simultaneously. Interactive based programming allows you to write program that

responds to the user actions and helps developers to just implement the logic based on the user action instead to manage the complete multi-tasking solution.

- **Architecture-Neutral:** The major challenge when Java was developing is to have programming language with which a program can be developed and executed anytime in future. With changing environments of hardware, processor, Operating system there was need to have program still adopt to this architecture changes. Java code does not depend on the underlying architecture and only depends on it JVM thus accomplish the architecture neutral programming language.

- **Interpreted:** The compiled code of Java is not machine instructions but rather its a intermediate code called Bytecode. This code can be executed on any machine that implements the Java virtual Machine. JVM interprets the Bytecode into Machine instructions during runtime.

- **High Performance:** When java programs are executed, JVM does not interpret entire code into machine instructions. If JVM attempts to do this then there will huge performance impact for the high complexity programs. JVM was intelligently developed to interpret only the piece of the code that is required to execute and untouched the rest of the code. The performance of java is never questioned compared with another programming language.

- **Distributed:** Java has a feature called Remote Method Invocation (RMI) using which a program can invoke method of another program across a network and get the output.

- **Dynamic:** Java programs access various runtime libraries and information inside the compiled code (Bytecode). This dynamic feature allows to update the pieces of libraries without affecting the code using it.