Applied Science & Engineering Journal for Advanced Research *Peer Reviewed and Refereed Journal*
ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

A Software Engineering Approach on Developing a Real Time Radar Target Generator for Airborne Targets

Ftoon Kedwan
Assistant Professor, Software Engineering Department, College of Computer and Cyber Sciences, University of Prince Mugrin, Medina, Saudi Arabia

Corresponding Author: ftoon.k19@gmail.com

Received: 10-02-2024	Revised: 27-02-2024	Accepted: 18-03-2024

ABSTRACT
This paper presents a novel method of a radar system testing in a simulated environment using an artificial Target Generator. This work follows the software engineering approach into implementing the radar software system. Waterfall software development life cycle model if adopted for the Target Generator radar system implementation.
The radar system can measure any type of radar target parameters supplied by a Radar Controller and can also transmit any radar signals generated by the Radar Target Generator. This work is essential for radar-related software development, testing, production, and maintenance.
If radar targets are to be tested using real flights, though it would produce accurate performance testing results, it would consume increased amounts of time and would be prohibitively costly and complex. Additionally, there is no guaranteed method of flying multiple flights with the same velocity, acceleration, direction, azimuth (angle), altitude, etc. The slight change in every flight affects the accuracy of the final testing results.
Hence, virtual radar targets are generated in real time via a virtual flight-testing environment to allow radar systems to be tested with flexible parameters' settings and high accuracy results. To accomplish this, aircraft dynamics, aircraft's Radar Cross Section (RCS), and other environmental effects are adjusted and customized in the virtual radar system testing environment. Then, the system will be ready to simulate as many flights as needed. This proposed software proved to be a faster and a more cost-effective solution. It is also programmable in any object-oriented programming language. In addition, due to the high modularity programming approach followed, this implementation is highly scalable and upgradable to any advanced deployment environment, such as governmental platforms.

Keywords: *radar target generator, software engineering, airborne targets, radar system*

I. INTRODUCTION

The term RADAR was first used around World War II by the United States Navy in 1940 as an acronym for Radio Detection And Ranging [1]. The radar systems are used in the modern world in a variety of ways. They are used in the health sector for public health surveillance as a range-controlled radar [2]. Additionally, they are used in the outer space missions as radar astronomy, spacecraft detection and tracing radar [3], outer space surveillance systems [4]. They are also used in the army and governmental defense as air-defense systems [5], landmarks locating and discovering marine radars, and antimissile and guided missile target locating systems [5, 6]. In addition, general uses of radar systems are used in organizing and managing aerospace crafts [7]. Such radar systems include terrestrial traffic control, and flight control and anti-collision systems [5, 8, 9, 10]. Other uses cover geological observations such as ground-penetrating radar, ocean surveillance systems, weather formations, terrain, motor vehicles, and meteorological precipitation monitoring [11].

Radar systems are used to detect objects' reflected radio waves or microwaves during their movement [5, 11]. Both transmit and receive radar process the radio waves parameters to determine object's properties such as velocity, azimuth, and range. The radar system operator can monitor these data off a Radar Display in a form of electromagnetic radiation pulses converted into a continuous electronic analog signal [11] which will be further processed and presented onto the screen display area. There are multiple radar display types depending on the scope of data presented and the computation ability. Some radar displays only display range data, some display range and azimuth data, and some of them display range, azimuth, elevation, and direction data [12].

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

II. LITERATURE SURVEY

Inverse Synthetic Aperture Radar (ISAR) [13] is a radar system developed to image airborne targets [14, 15]. Ground-to-Air Imaging Radar (GAIR) system [10] can collect ISAR image data in addition to the measurement data of the Total Radar Cross Section (TRCS) [16]. The GAIR system performance is distinguished due to the use of a calibrated, transportable, and remotely operatable radar system.

Virtual radar echo signals were being generated via fiber optic delay lines [17, 18]. Signals capture and replay were handled by the digital radio frequency memories (DRFMs) [19]. More recently, to simulate radar targets, signal generators and analyzers are used [7, 20, 21]. This evolution is mainly to make sure the radar system is successfully functioning as per user requirements. Therefore, more affordable measurement equipments are being developed in this paper for the radar system components to be tested, verified, measured, and validated against user requirements.

Target Audience

The target audience of the current work could be radar-related project implementation managers or investors, government agencies, end-users, etc. For example:

- Navy (Airborne Radars, Coastal Security Radars)
- Army (Over the Horizon Radars, Missile and Gunfire Control Radars, Long Range Surveillance Radars)
- Air Force (Precision Approach Radar, Airborne Radars)
- Radar Testing Agencies

Game developers can also be possible stakeholders for such technology. The famous 'War Thunder' video game uses a similar implementation.

Scope and Objectives

A virtual radar system testing environment can test and evaluate the recognition and the identification of hundreds of simulated airborne radar targets in an hour. For each run, the user can apply the exact same flight parameters, or can change them slightly or significantly. This setting can never be done easily on a realistic physical flight-testing scenario. Even when it is made possible, it cannot be repeated hundreds of times with a comparable accuracy, efficiency, and cost and time efficiency as is the case with virtual testing environments. Let alone the need to test the radar system many times for every maneuvering mode and airborne target type.

III. IMPLEMENTATION AND METHODOLOGY

Software Requirements and Specifications

An automatic Real Time Radar Target Generator software will be developed which interacts with a Radar Controller. This interaction occurs through User Datagram Protocol (UDP) [22] for sending requests and receiving computed data. Hence, the Target Generator supports UDP communication through which requests will be received, and computed data will be sent in predefined formats.

The proposed software (Real Time Radar Target Generator) is able of loading the scenarios generated by the Scenario Modeler and perform various computations on the scenario data with high flexibility to different target entities (i.e., helicopter, directed missiles) and number of segments (maneuvering tracks).

Radar Data Processor (RDP) [23] is one of the critical Radar System components. RDP receives and processes signals and plots from other radar components. RDP can filter out noise in the received data. Despite any exceptional or extreme maneuvering situations, RDP is able of initiating and updating tracks with their specific parameters with high accuracy. RDB then displays those tracks on an Air Situation Display [12].

The proposed solution in the current paper uses the Plan Position Indicator (PPI) [24] as a Radar Display. The PPI receives echo from a 360-rotating radar antenna displayed in the center with surrounded circles used to draw curves representing the distance and the height of detected target objects [24].

The Radar Target Scenario Modelling [25] software (a.k.a. Scenario Modeler), which will also be developed, specifies the detection criteria for any simulated radar target entity (fighter, defense, transport, or commercial aircrafts). In addition, the system specifies the flying conditions and the maneuvering type of that entity in regard to velocity, acceleration, turning, and climbing. To test the effectiveness of this Scenario Modeler, the Radar Data Processor performance will be evaluated and tested.

The software shall have a user-friendly Graphical User Interface (GUI) (a.k.a. Graphical Editor) for displaying the scenarios on a PPI diagram with zooming capability. It should be able to connect to a host IP address with configurable send and receive port numbers. The software shall wait at the receive port to get requests sent by the Radar Controller, process the

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

request to generate the requested target parameters and then send the computer parameters back to the send port for the Radar Controller to read it. This process should run seamlessly without freezing until the Radar Controller is disconnected. The GUI shall support a function to save scenarios as images and also print those scenario images.

Due to the lack of an actual Radar Controller during the development of the current project, a driver application capable of generating, sending, and receiving requests through a UDP channel has to be developed as well.

User Requirements

The software should have a Main GUI with File and Tools Menu tabs. File Menu should have Open, Close, Print and Exit options. The Tools Menu should have Connect and Save Image options. There should be a Toolbar with Open, Print, Save and Connect buttons.

Through the Open menu button, the user should be able to open and load a saved or a predefined scenario file. Once chosen, the scenario described in this file should get displayed in the PPI display area.

A scenario is composed of multiple targets. Each target is defined as a sequence of interconnected tracks. Each track exhibits its own maneuvering characteristics. Clicking the connect menu button shows a wizard for starting a UDP connection. The wizard allows the user to set Host IP Address and Send and Receive Port Numbers. The Radar Controller will request the targets' time, azimuth and elevation parameters to be identified and computed by the system. Then, the targets' acquired and computed parameters have to be sent to the send port. The computed parameters include target ID, coordinates, altitude, range, azimuth, elevation, heading direction, time, acceleration, and velocity.

The below list summarizes the important user requirements:

- Load Scenario
- Print Scenario
- Close Scenario
- Exit operation
- Save Scenario as an Image
- Connect via UDP
- Receive Request via UDP
- Compute Target Parameters
- Send Target Parameters via UDP
- PPI Display
- Zooming in and out
- Status Bar with Status display

System Requirements

System hardware requirements:

- System Type: PC or Workstation.
- Processor: Multi-core Intel Core i3/i5/i7/Xeon x86/x64 with at least 2.0 GHz clock speed.
- Primary Memory ≥ 3 GB.
- Graphics Card: NVIDIA Quadro/GeForce with at least 2 GB Graphics Memory.

System software requirements:

- Operating System: Microsoft Windows 7 or later.
- Development Framework: QT 5.0 or later.
- Programming Language: C++ 11.
- Compiler: Microsoft Visual Studio 2013 or later.

IV. DESIGN AND ARCHITECTURE

Use Case Model

To analyze the developed system, a use case diagram is described in Fig. 1 to explain system requirements.

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

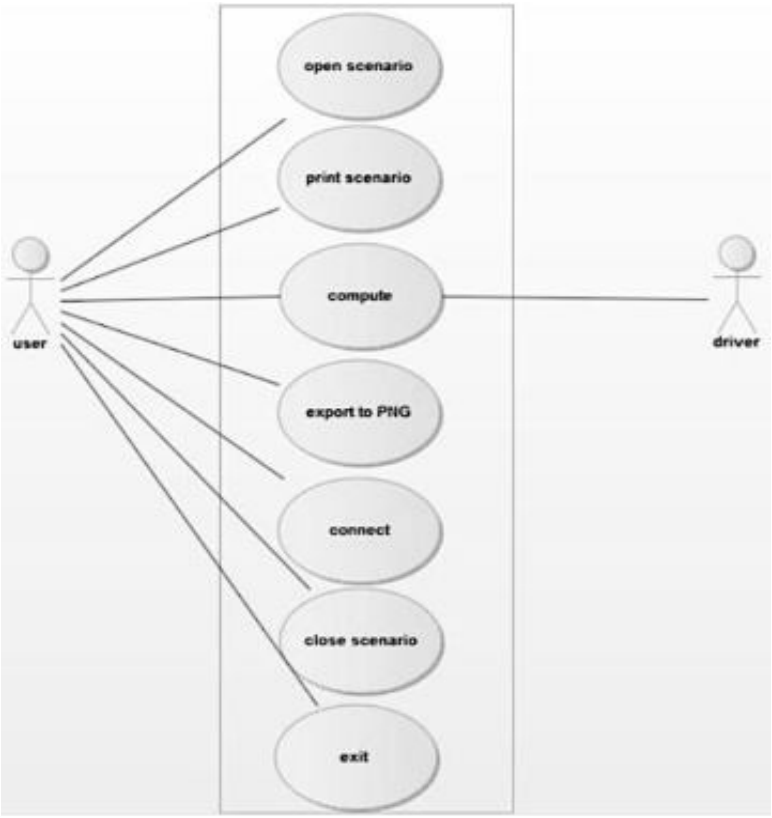


Figure 1: Use Case Model

Use Case Descriptions

Two of the main use case scenarios are described below:

Use Case: Open Scenario

Criteria	**Description**
Purpose	Opening a scenario file to be loaded on the display screen.
Pre-Conditions	System main window is displayed.
Post-Conditions	The display screen is showing the selected scenario.
Use Case Flow	User clicks on the system main menu: File-> Open A window opens to choose an XML file. User clicks on a scenario file, then, user clicks open. The chosen scenario file is displayed on the display screen.

Use Case: Close Scenario

Criteria	**Description**
Purpose	Closing an opened scenario file.
Pre-Conditions	A scenario file is already displayed on the display screen.
Post-Conditions	The display screen is empty, and any scenario file is closed.
Use Case Flow	User clicks on the system main menu: File-> Close The scenario file disappears from the system display area.

System Sequence Diagram

The sequence diagram is one of the UML (Unified Modeling Language) standard notations used to demonstrate developed systems. It presents the interaction between the system objects or subcomponents. It also shows the messages sent back and forth during the interaction episodes with respect to time sequence.

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

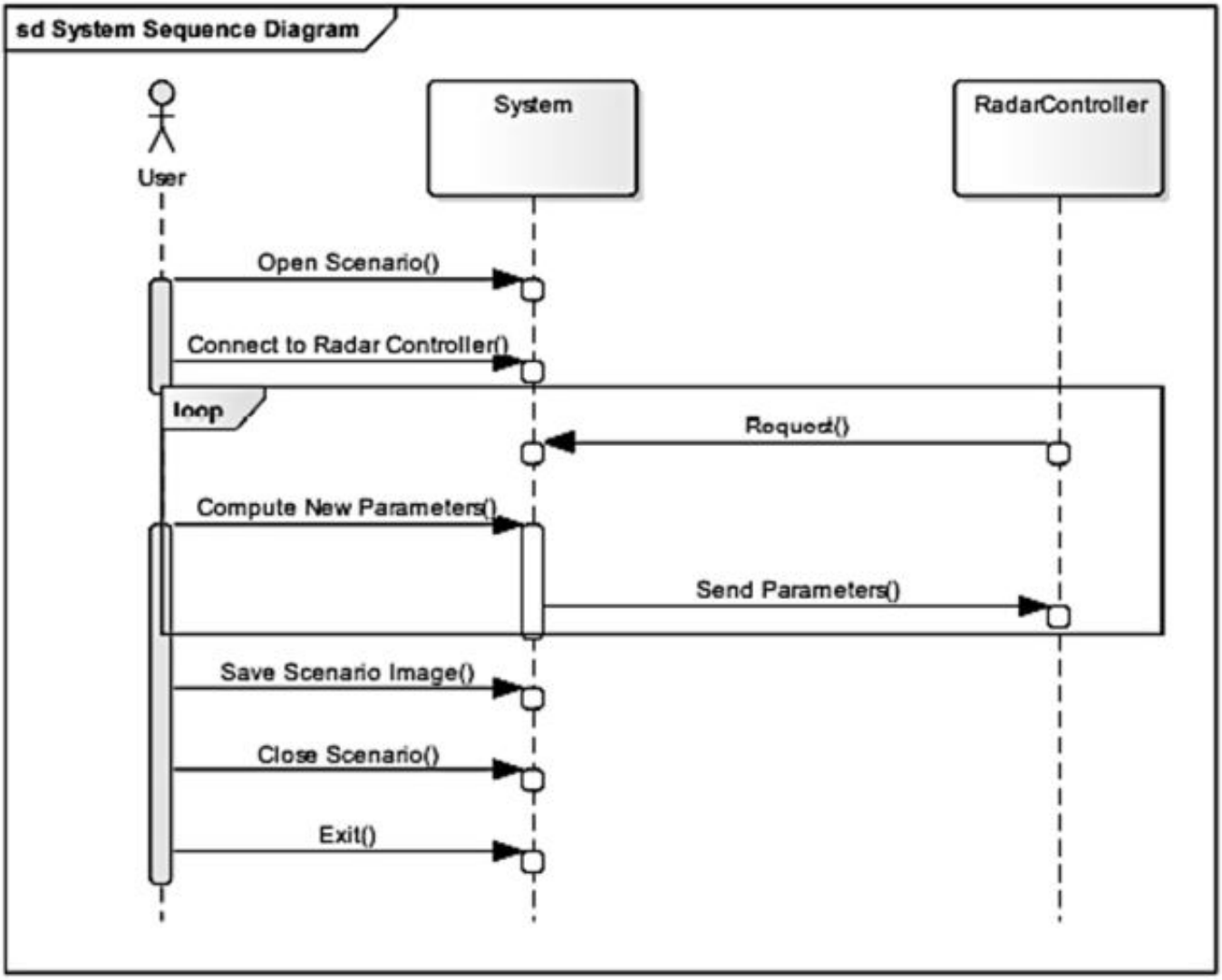


Figure 2: System Sequence Diagram

The sequence diagram in Fig. 2 summarizes a successful interaction between a user and the developed system. The roles enacted by the main actors, the user and the Driver (Radar Controller), identified in the use case scenario, Fig. 1, are chronologically presented in Fig. 2. In what follows, two of the main use case scenarios (Open and Connect) are presented in sequence diagrams.

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

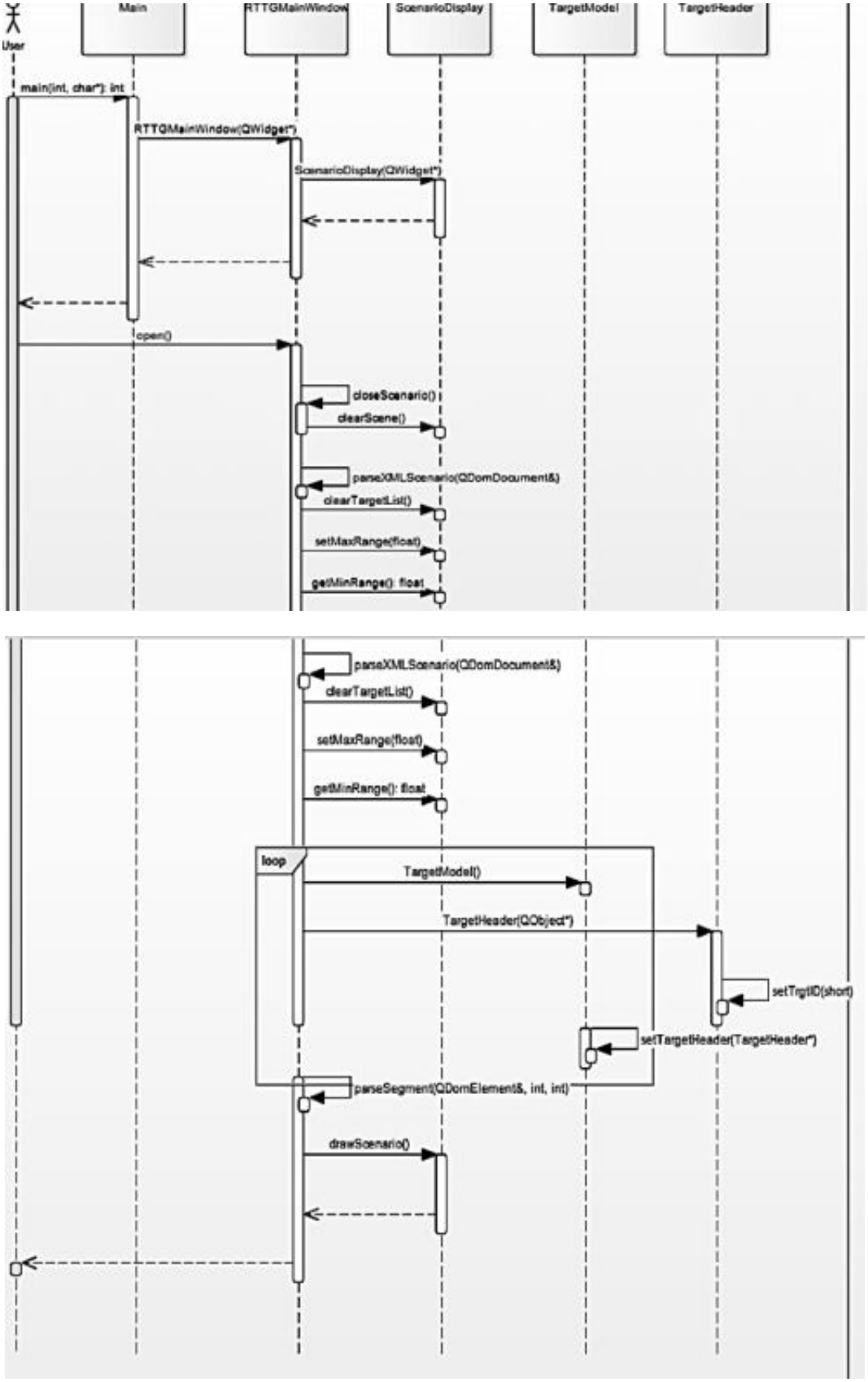


Figure 3: Open Scenario Sequence Diagram

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

Open Scenario

The sequence diagram in Fig. 3 demonstrates the open use case scenario with the following operations executed:

- User calls the main function.
- Main function calls the MainWindow as it encapsulates it.
- MainWindow calls ScenarioDisplay() to draw and display the scene, and then close it.
- From MainWindow, the user calls for the open() operation.
- MainWindow first closes all of the already opened scenarios.
- parseXMLScenario will parse the newly selected scenario.
- MainWindow will call the clearTargetList() operation to clear the previously executed list of targets.
- MainWindow will set max and get min range.
- MainWindow then enters into the loop in which it calls the target model and generates its ID.
- Once target ID is set, MainWindow will set the target header and then call the parse segment.
- Lastly, MainWindow will draw the corresponding scenario.

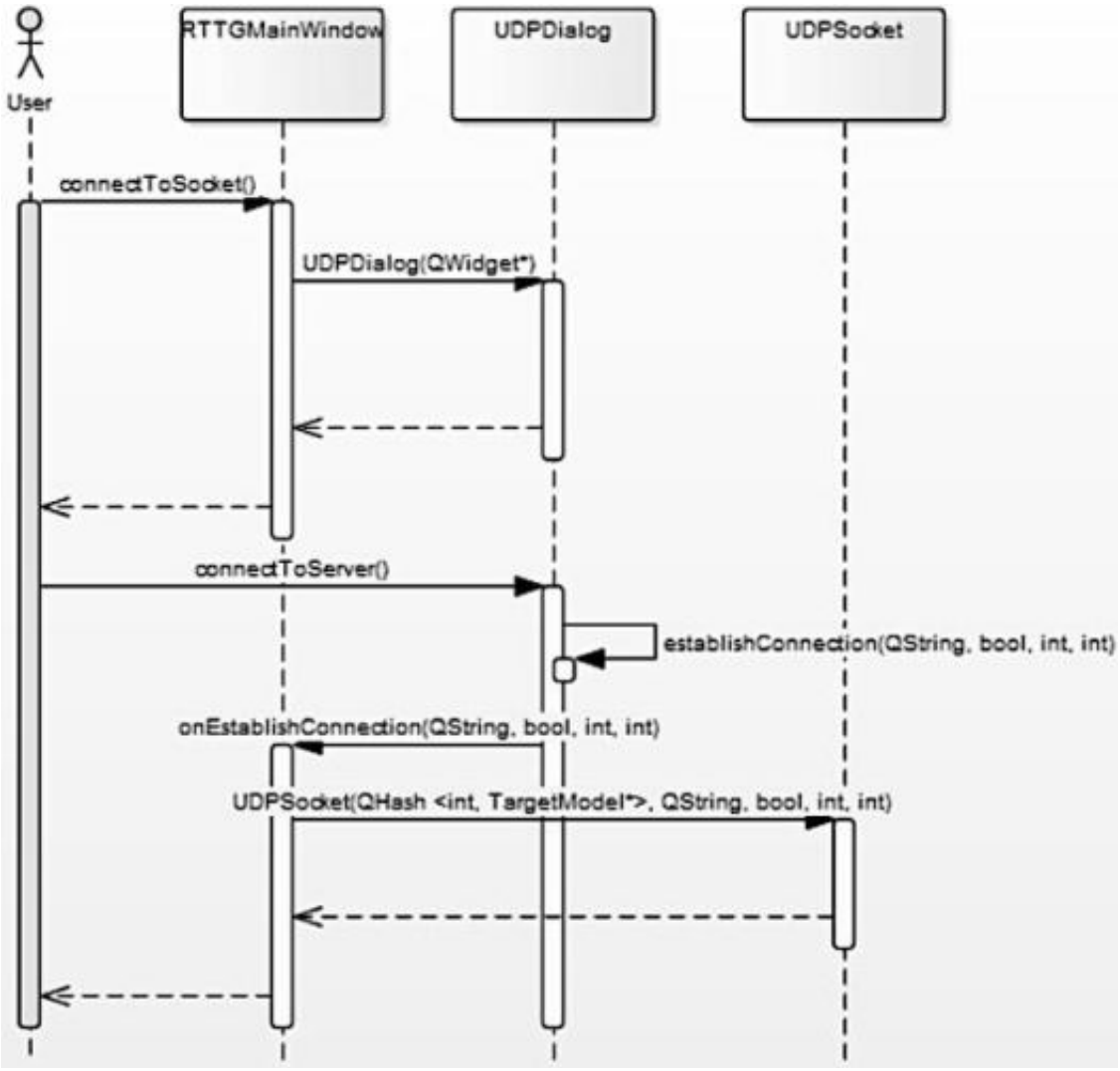


Figure 4: Connect Scenario Sequence Diagram

Connect Scenario

The sequence diagram in Fig. 4 demonstrates the connect scenario with the following operations executed:

- User interacts with the MainWindow to conduct socket connection.
- MainWindow calls the UDPDialog function.

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

- User requests connection to the UDPDialog and reads the fields provided.
- UDPDialog establishes a connection and returns the values achieved.
- User conducts a connection with UDPDialog as it wants to communicate with the server.
- UDPDialog starts the estalishconnection() function to process the request and then call on establishconnection() from MainWindow.
- MainWindow will generate the connection and return back to the UDPSocket and calls the UDPSocket() function.
- At the end, all the values will be returned to the user.

High Level Architecture

Fig. 5 identifies the system interfaces responsible for processing user-system interaction and data transfer.

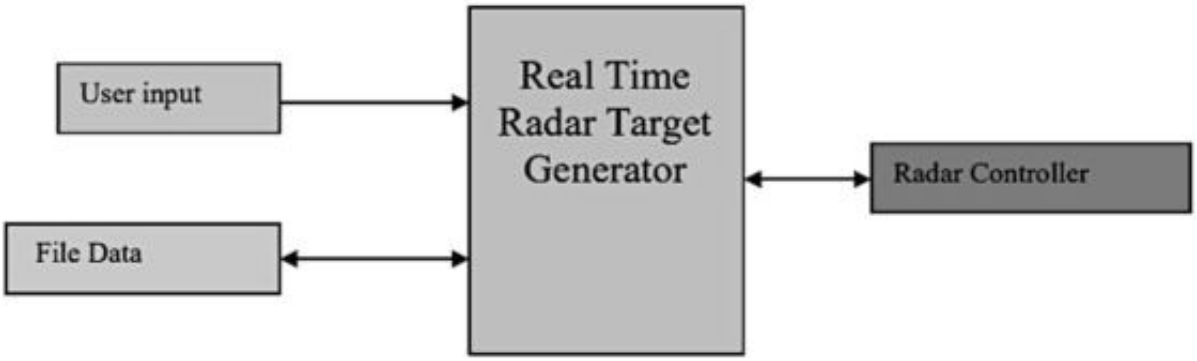


Figure 5: System Interfaces

The system architecture is shown in Fig. 6. There is a Main Module which creates the master GUI and controls the entire operations. The Display Module displays a PPI diagram where the loaded scenarios will be displayed. Target Model captures all received target parameters such as velocity and azimuth parameters, and the target's list of tracks or segments.

Every track or segment has its elevation, azimuth, and velocity parameters, in addition to the start and end positions. The segment module handles all of the computations as per various maneuveringmodes. The communication module sends and receives requests from the Radar Controller through UDP communication over a network. The Driver module acts as a substitute for the Radar Controller and sends requests to the Target Generator through UDP communication. The results received will be displayed on the system console.

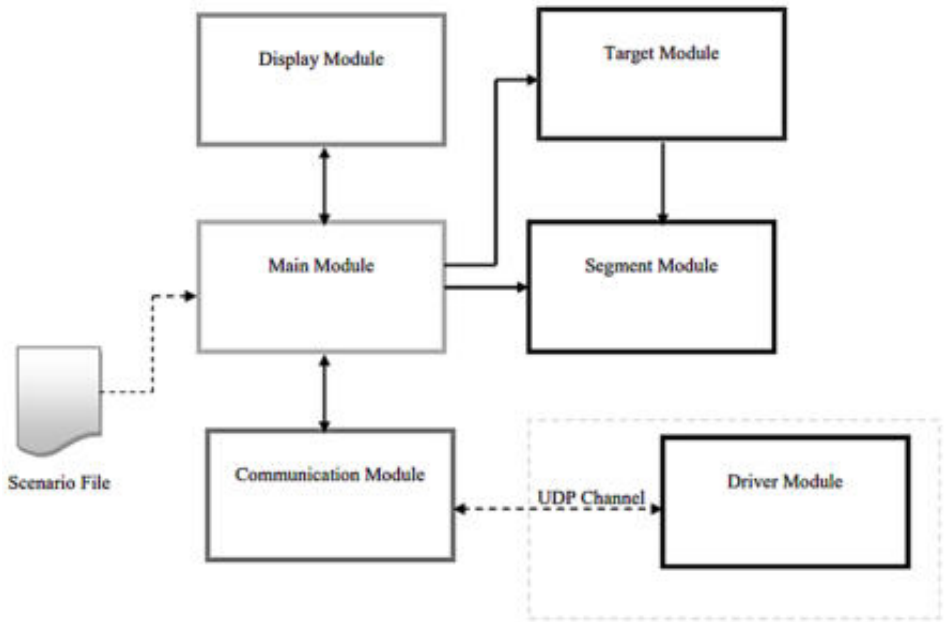


Figure 6: System Architecture

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

V. IMPLEMENTATION

Class Model

To further describe the structure of the developed system, a UML class diagram is used in Fig. 7. The class diagram shows the relationships among the classes where each class represents a separate object. For a better readability, each class is further expanded with their defined attributes and methods details in the appendix, in appendices 1-13.

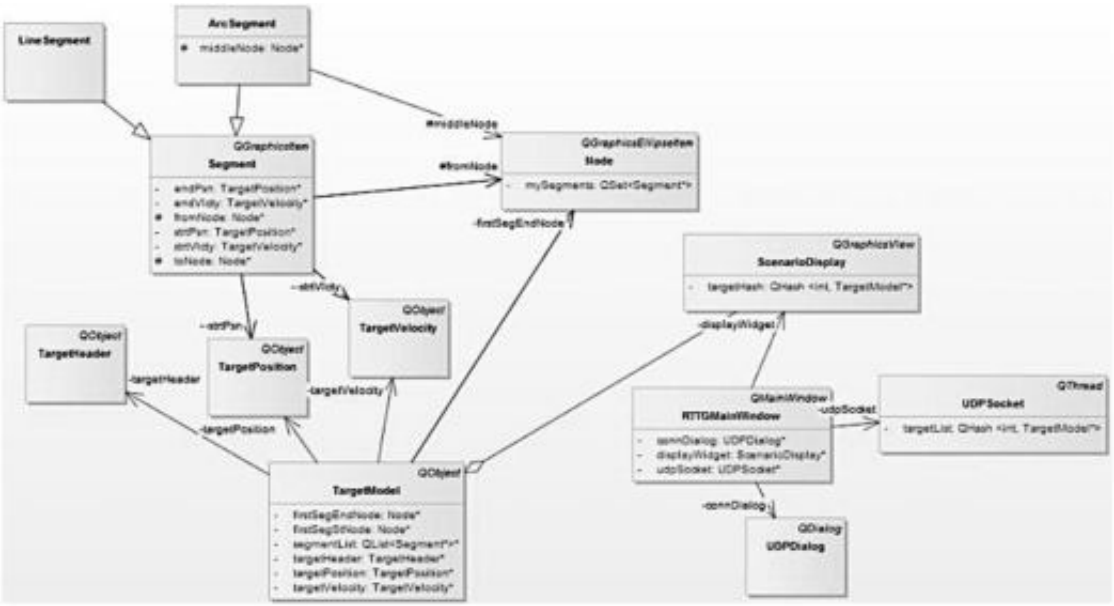


Figure 7: Complete System UML Class Diagram

Development Plan

Fig. 8 shows a data flow diagram representing the development plan of the Radar Target Generator system. The solid lines in the diagram are for data flow, while dashed lines represent control flow.

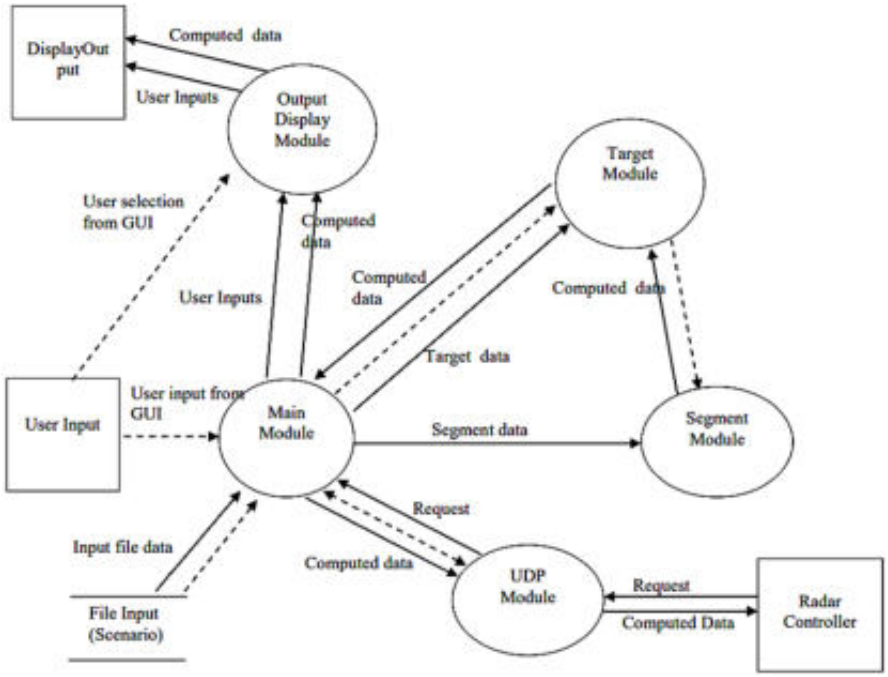


Figure 8: Radar System Data Flow Diagram

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

The following is a description of the main system components depicted in Fig. 8:

- Main Module: The Main interaction point between the system and the user through GUI menus and tool bars.
- Target Module: All captured or computed target parameters are stored in this module.
- Segment Module: All captured or computed segment parameters are stored in this module, where each segment represents a specific flight maneuvering mode.
- UDP Module: The User Datagram Protocol (UDP) is in charge of sending and receiving target parameters from the Radar Target Generator to the Radar Controller (Driver) in real time uninterruptedly.
- Radar Controller (Driver): The Radar Target Generator receives the requested target parameters from the Driver, which acts in place of the Radar Controller in this development work. Once a target has been identified to be matching with the requested parameters, the target's parameters are computed and sent to the radar system.
- Output Display Module: A scenario visualization display with captured target details (i.e., current target's position and elevation) shown in the system console.

Development Framework

- Since the software requirements are completely defined, waterfall model will be used as the Software Development LifeCycle (SDLC) Model.
- For development, QT framework and Microsoft Visual Studio compiler will be used for their GUI aspects and compatibility with different operating systems.
- QT creator and designer Integrated Development Environment (IDE) are employed for developing GUI interfaces.
- The programming language used is C++ 11.
- UML modelling language will be used for an object-oriented approach.
- A Work Breakdown Structure (WBS) will be made for better project management.
- Unit tests will be performed at class/module level, followed by integration and system level tests.

To develop the Radar Target Generator system software, QT software framework is used. QT supports cross-platform development [26] of software with a GUI. The QT compilers used in the current paper are the GCC C++ compiler and the Visual Studio suite.

QTQuick declarative scripting language (QML) is used for coding logic in JavaScript. Other QT used tools include XML and JSON parsing, thread management, SQL database access, and network support. To communicate between the system objects, mainly the GUI and the server software components, QT signals and slots are used to handle events. Event information are sent as signals whereas slots receive them.

Visual Studio Runtime Environment is used to run the software. The resources and executables were copied into the environment and executed. The system could also be built as an application and executed using the .exe file.

A couple of computation algorithms were needed in this software development for computing azimuth, heading direction, elevation, ground range, polar to cartesian conversion, cartesian to polar conversion, velocity, altitude, and detection probability [27]. In addition, segment duration must be computed considering the current maneuvering mode and kinematic conditions besides other affecting parameters.

To encourage the development of similar systems, most of the system components were built as reusable software components. The coding phase followed a high modularity approach considering seamless user-system interaction.
Appendices 14-26 demonstrate the developed system components and the GUI screens and interfaces.

VI. TESTING

In order to test the developed software, the following set-up was followed:

- The Driver (Radar Controller) is set to pass static beam information (i.e., azimuth and velocity).
- Only the time parameter changes while the radar looks continuously in one direction.
- The covered azimuth range is from -100 to +100.
- The covered elevation range is from 1.50 to 5.50.
- Requests were sent continuously from the Driver to the Target Generator through UDP while both system components are hosted in the same workstation.

Different scenario cases were created for testing the Radar Target Generator software. The situations and the corresponding outputs are shown in Figures 9-16, where the left-hand figure is the actual scenario and the right-hand one is during the processing of the requests from the server. It can be observed that the computed positions (shown in red) are perfectly matching with the actual track (shown in yellow).

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

1. One Target Flying Steadily with Constant Velocity

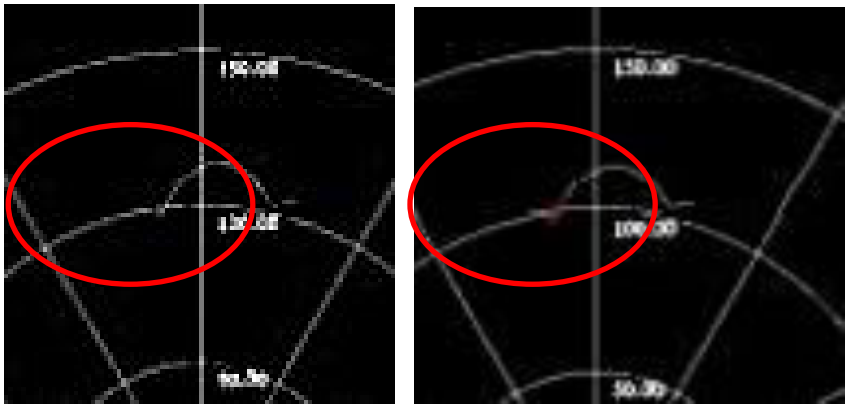


Figure 9: Single Target - Steady Flight

2. One Target Attempting a Level Turn

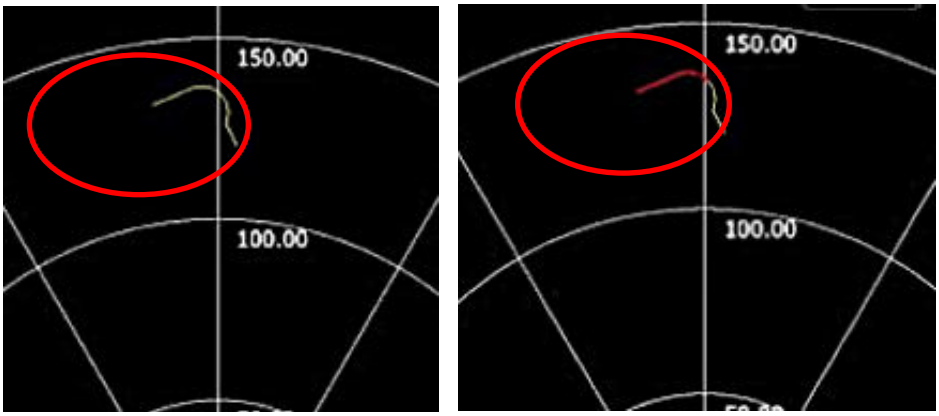


Figure 10: Single Target - Level Turn Flight

3. One Target Attempting an Upward Climb

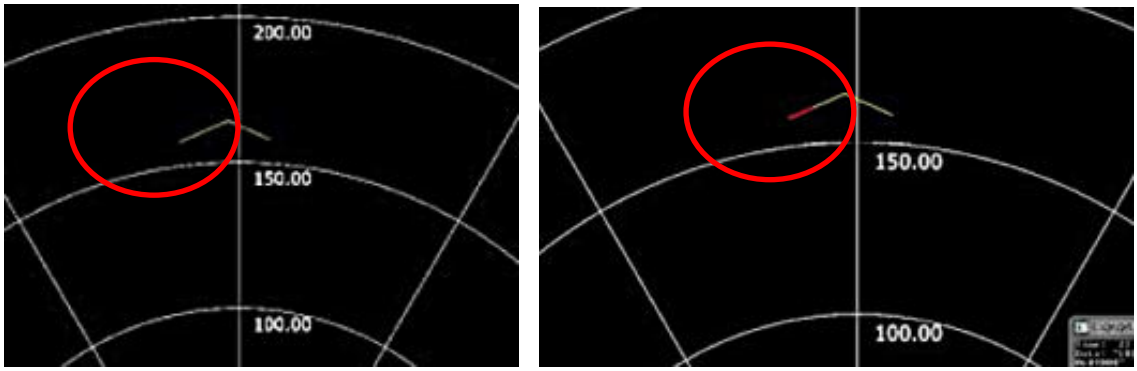


Figure 11: Single Target - Climb Flight

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

4. One Target Attempting an Accelerated Level Turn

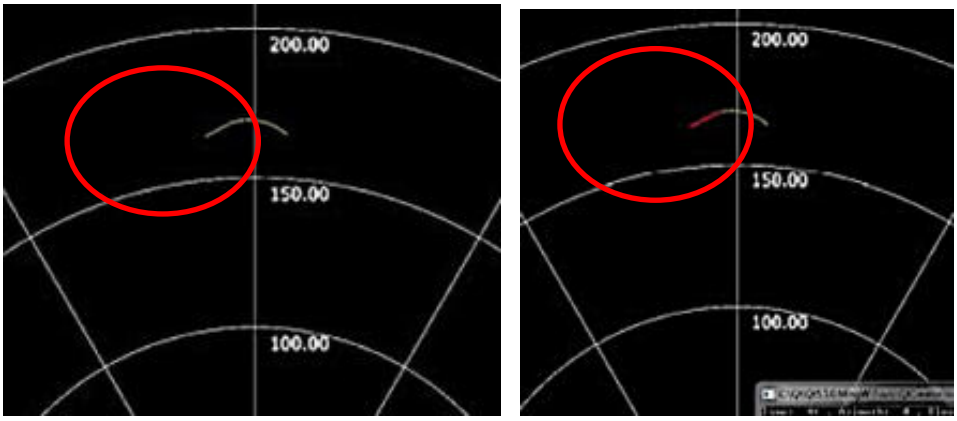


Figure 12: Single Target - Turn and Accelerate Flight

5. One Target Attempting a Turn with an Upward Climb

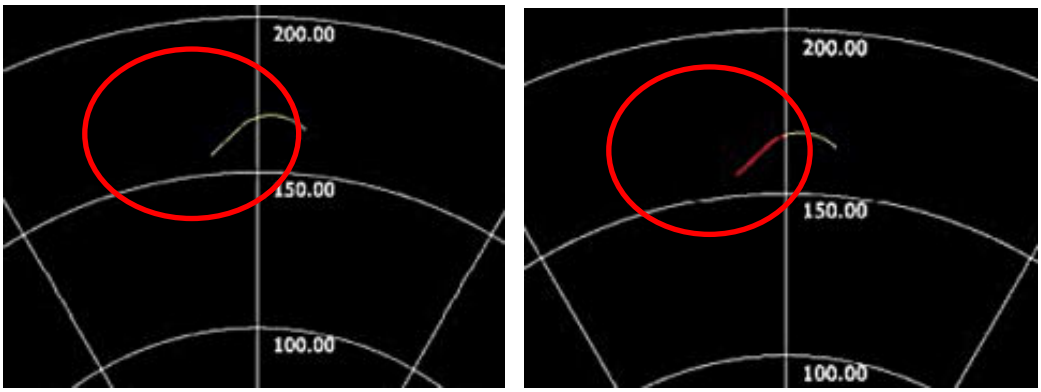


Figure 13: Single Target - Turn and Climb Flight

6. One Target Attempting All Maneuvering Modes

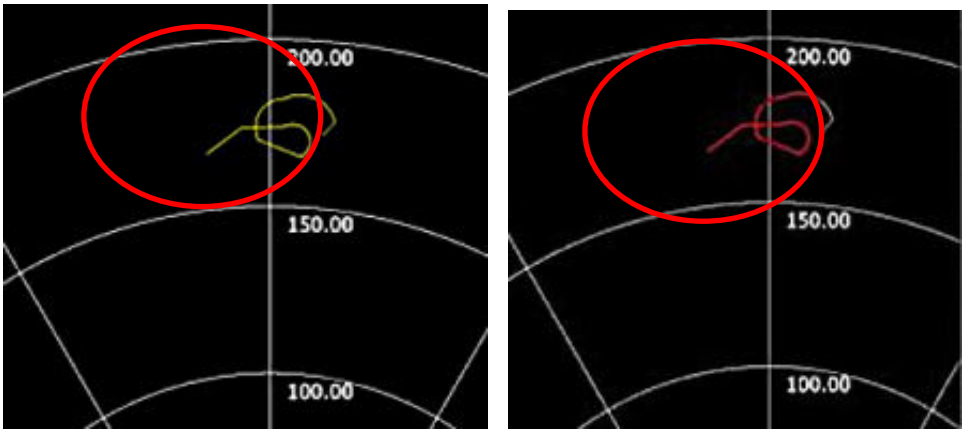


Figure 14: Single Target - All Maneuvering Modes

In the left-hand scenario, the target flies in a steady mode in the first segment, switched to accelerated mode in the second segment. In the third segment, the target attempts a level turn mode, to climb mode inthe fourth. In the fifth segment, the target accelerates and then turns and climbs in the sixth segment.

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

7. Two Targets Flying Steadily with Constant Velocity

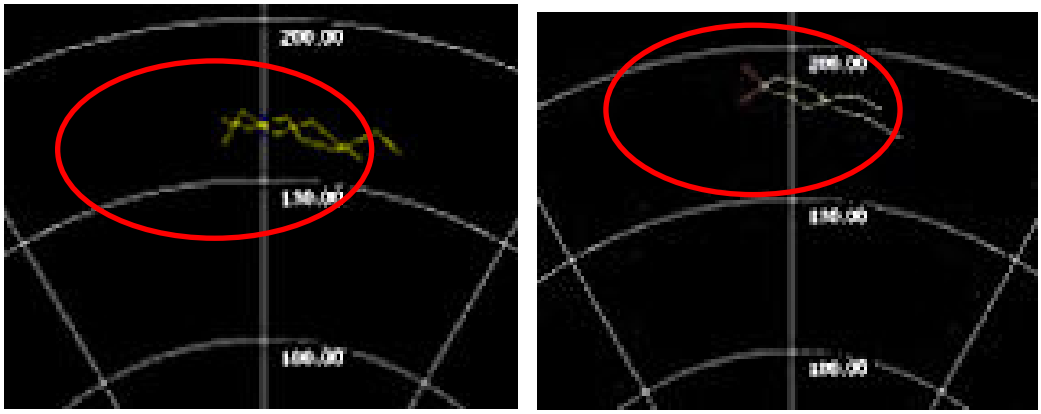


Figure 15: Multiple Targets - Steady Flight

8. Two Targets, One is Flying Steadily with Constant Velocity and the Other Flying in an Accelerated Mode in Two of the Track Segments

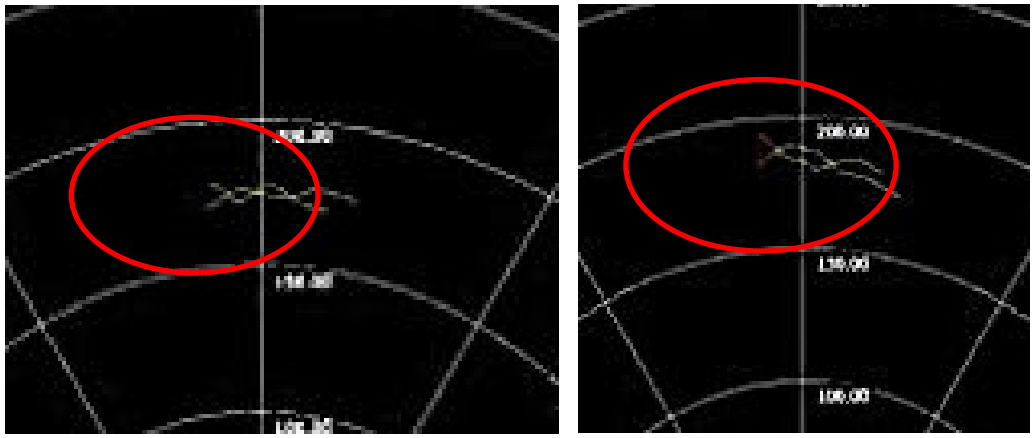


Figure 16: Multiple Targets – Accelerated Flight and Steady Flight

The synthetic environment of the radar system, with its subcomponents (Radar Target Generator, Radar Controller, Radar Data Processor, Scenario Modeler, Plan Position Indicator), was successfully tested for its functionality, accuracy, and performance efficiency. The testing undergone different target maneuvering conditions. After the process of manual testing, it is found that the software has met all designated requirements and no bugs were discovered.

VII. DISCUSSION

There could be possible risks involved in implementing such a project as it is a complicated project to be rebuilt and or reimplemented. Some of the foreseen risks involved could be:

- User requirements are not fully captured or user ignorance.
- Wrong interpretation of user requirements.
- Inadequate testing performed.
- Schedule over-run.
- Inadequate previous experience in the problem domain.
- Lack of sufficient programming knowledge.

Some of those risks could be overcome through:

- Literature survey could overcome the lack of domain experience to some extent.
- Frequent interaction with stakeholders and project managers assures the conformance with user requirements.
- Division of modules amongst team members and parallel implementation contributes to on-time delivery.

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

- Studying similar implementations.
- Early planning and allocation of sufficient time for testing.

VIII. CONCLUSION AND FUTURE WORK

Conclusion

Developing a radar target generator software is necessary to help radar systems testing with limited budgets and restricted time frames. State of the art software components were included in the development and implementation phase. The radar system was implemented successfully with high modularity, scalability, and upgradability. Integration and system testing yielded promising results in terms of accuracy, efficiency, and overall functionality and performance. A software engineering approach was followed in this paper to develop a radar target generator software system. This paper can be used as a road map for any software engineering project or research work [28].

Future Work

- Integrating the Radar Display with a Geographic Information System (GIS) map to provide a more interesting and improved radar target view.
- Including other communication mechanisms than the UDP communication, such as TCP/IP communication protocols, to communicate with the Radar Controller.
- Automatic smoothening of scenario maneuvers from one segment to another for a more realistic scenario view.
- Automatic segment parameters re-computation when a scenario maneuver is changed.
- Merge the use of artificial intelligence and neural networks for a built-in radar target testing and verification [29].

REFERENCES

1. Sarkar, T. K., Palma, M. S., & Mokole, E. L. (2016). Echoing across the years: A history of early radar evolution. *IEEE Microwave Magazine*, *17*(10), 46-60.
2. Graichen, C., Ashe, J., Ganesh, M., & Yu, L. (2012, December). Unobtrusive vital signs monitoring with range-controlled radar. In: *2012 IEEE Signal Processing in Medicine and Biology Symposium (SPMB),* pp. 1-6. IEEE.
3. Ly, H. D., & Liang, Q. (2007, October). Collaborative multi-target detection in radar sensor networks. In: *MILCOM 2007-IEEE Military Communications Conference*, pp. 1-7. IEEE.
4. Cataldo, D., Gentile, L., Ghio, S., Giusti, E., Tomei, S., & Martorella, M. (2020). Multibistatic radar for space surveillance and tracking. *IEEE Aerospace and Electronic Systems Magazine*, *35*(8), 14-30.
5. Heuel, S., & McCarthy, D. (2015). Real time radar target generation. *Microwave Journal*, 92-106.
6. Vick, A., Zeigler, S., Brackup, J., Meyers, J. S., & Rand corporation. (2020). *Air base defense: Rethinking army and air force roles and functions.* RAND Corporation, pp. 0161.
7. Kavyashree.V, P.N. Madhu Chaitra, Prasad. A.Y, & Vinutha.H. (2017, May). A radar target generator for airborne targets. *IJSTE - International Journal of Science Technology & Engineering, 3*(11).
8. Fan, Y., Xiang, K., An, J., & Bu, X. (2013, April). A new method of multi-target detection for FMCW automotive radar. In: *IET International Radar Conference,* pp. 1-4. IET.
9. Moses, A. A. (2013). *Radar based collision avoidance for unmanned aircraft systems*. Doctoral Dissertation, University of Denver.
10. Goodman, R., Nagy, W., Wilhelm, J., & Crippen, S. (1994, March). A high-fidelity ground to air imaging radar system. In: *Proceedings of 1994 IEEE National Radar Conference,* pp. 29-34. IEEE.
11. Yanovsky, F. (2008). Millimeter-wave radar: principles and applications. In: *Millimeter Wave Technology in Wireless PAN, LAN, and MAN*, pp. 315-386. Auerbach Publications.
12. Bair, G. L. (1996). *Airborne radar simulation*. Camber Corporation, Dallas, Texas.
13. Prickett, M. J., & Chen, C. C. (1980). Principles of inverse synthetic aperture radar/ISAR/imaging. In: *EASCON'80; Electronics and Aerospace Systems Conference,* pp. 340-345.
14. Liu, J., Xie, J., Nie, Y., Wang, Y., & Cui, G. (2021, May). A new airborne radar target detection approach based on conditional generative adversarial nets. In: *2021 IEEE 4th International Conference on Electronics Technology (ICET),* pp. 184-188. IEEE.
15. Moore, S. (2009, October). UK airborne AESA radar research. In: *2009 International Radar Conference" Surveillance for a Safer World"(RADAR 2009)*, pp. 1-7. IEEE.
16. Borkar, V. G., Ghosh, A., Singh, R. K., & Chourasia, N. K. (2010). Radar cross-section measurement techniques. *Defence Science Journal*, *60*(2), 204.

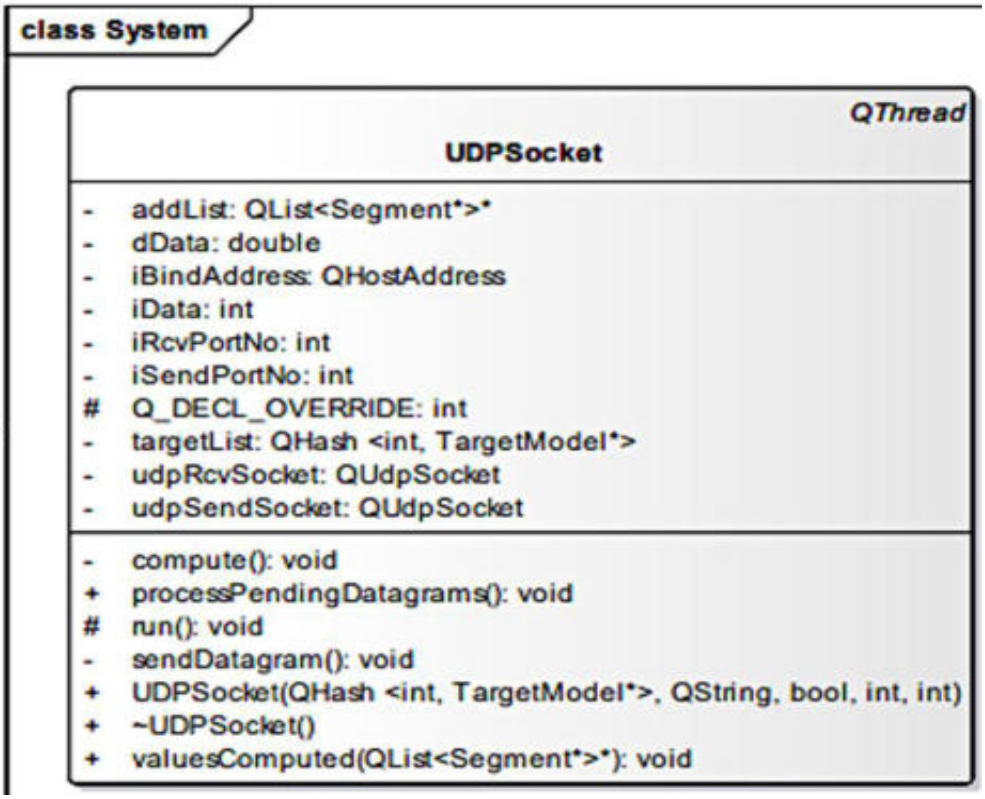
ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

17. Prcic, M. M. (1994). Flyable fiber-optic radar target generator. *IEEE Aerospace and Electronic Systems Magazine*, *9*(1), 17-20.
18. Xiuwei, C., Yunhua, Z., & Xiangkun, Z. (2009, October). Radar echo simulation system with flexible configuration. In: *2009 2nd Asian-Pacific Conference on Synthetic Aperture Radar,* pp. 365-368. IEEE.
19. Berger, S. D. (2003). Digital radio frequency memory linear range gate stealer spectrum. *IEEE Transactions on Aerospace and Electronic Systems*, *39*(2), 725-735.
20. Sagayaraj, M. J., Jithesh, V., Singh, J. B., Roshani, D., & Srinivasa, K. G. (2018). A hybrid approach to cognition in radars. *Defence Science Journal*, *68*(2), 183.
21. Heuel, S. (2015, June). Radar target generation. In: *16th International Radar Symposium (IRS),* pp. 1002-1009. IEEE.
22. Postel, J. (1980). User datagram protocol. *The RFC Series*. (No. rfc768).
23. You, H., Jianjuan, X., & Xin, G. (2016). *Radar data processing with applications*. John Wiley & Sons.
24. Williams, F. C., Howell, W. D., & Briggs, B. H. (1946). Plan-position indicator circuits. *Journal of the Institution of Electrical Engineers-Part IIIA: Radiolocation*, *93*(7), 1219-1255.
25. Orman, A. J., Shahani, A. K., & Moore, A. R. (1998). Modelling for the control of a complex radar system. *Computers & Operations Research*, *25*(3), 239-249.
26. Kedwan, F., & Sharma, C. (2019). Model-driven software development platforms reviews. *International Journal of Computer Applications*, *178*(31), 24-33.
27. Lee, H. B. (2016). *Efficient parameter estimation methods for automotive radar systems.* Doctoral Dissertation, Department of Electrical and Computer Engineering, College of Engineering, Seoul National University.
28. Kedwan, F. (2023). Traffic light timer adjustment system using threshold edge detection for image processing: A software engineering approach. *International Journal of Current Research in Multidisciplinary (IJCRM*), *8*(3), 09-36. doi:10.56581/IJCRM.8.3.09-36.
29. Kedwan, F., & Sharma, C. Twitter texts' quality classification using data mining and neural networks. *International Journal of Computer Applications*, *975*, 8887.

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

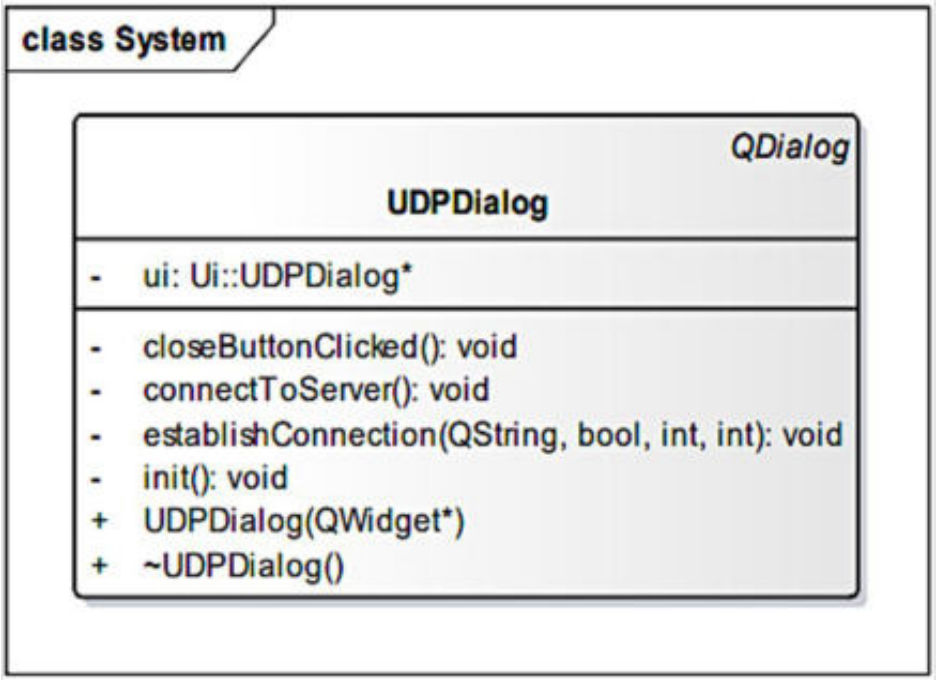
APPENDIX

Appendix 1



Class 1: UDPSocket Class

Appendix 2



Class 2: UDPDialog Class

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

Appendix 3

Class 3: MainWindow Class

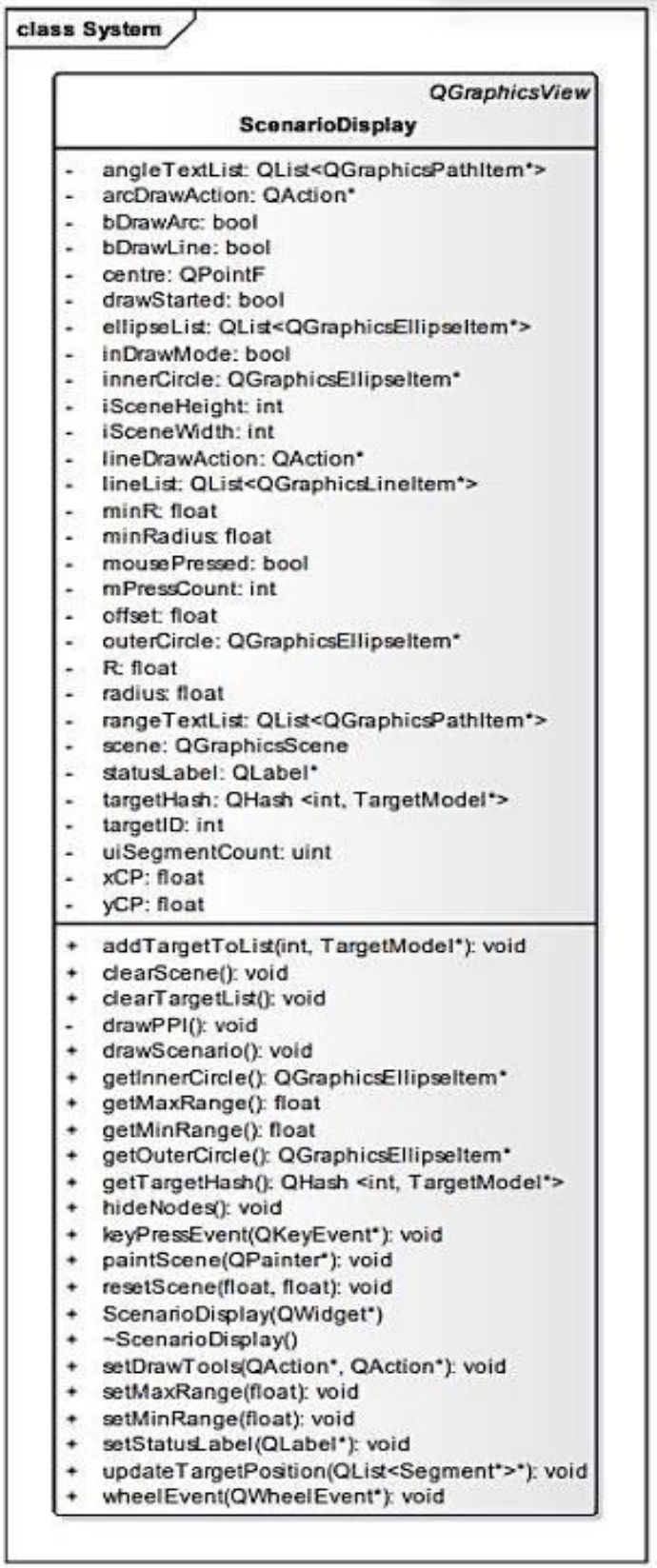
ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

Appendix 4

Class 4: Segment Class

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

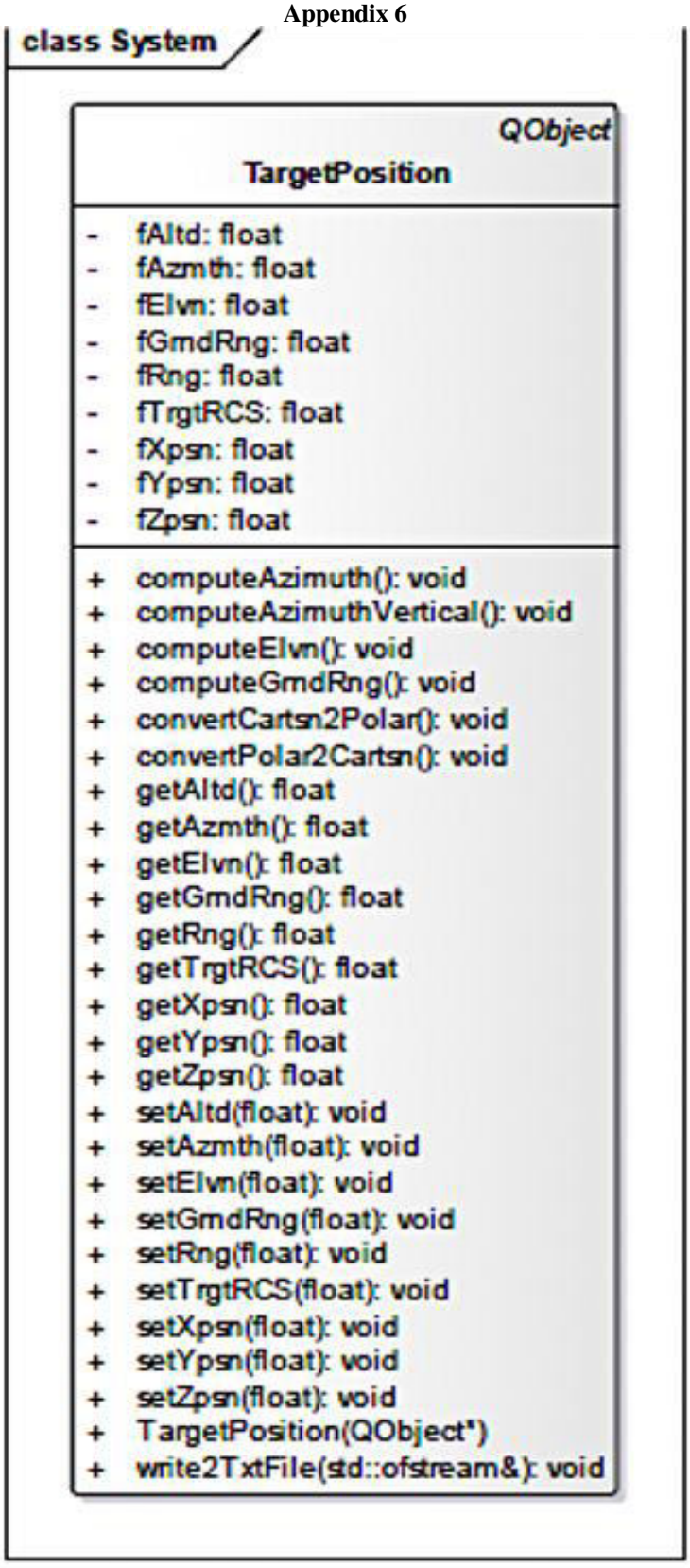
Appendix 5



Class 5: ScenarioDisplay Class

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

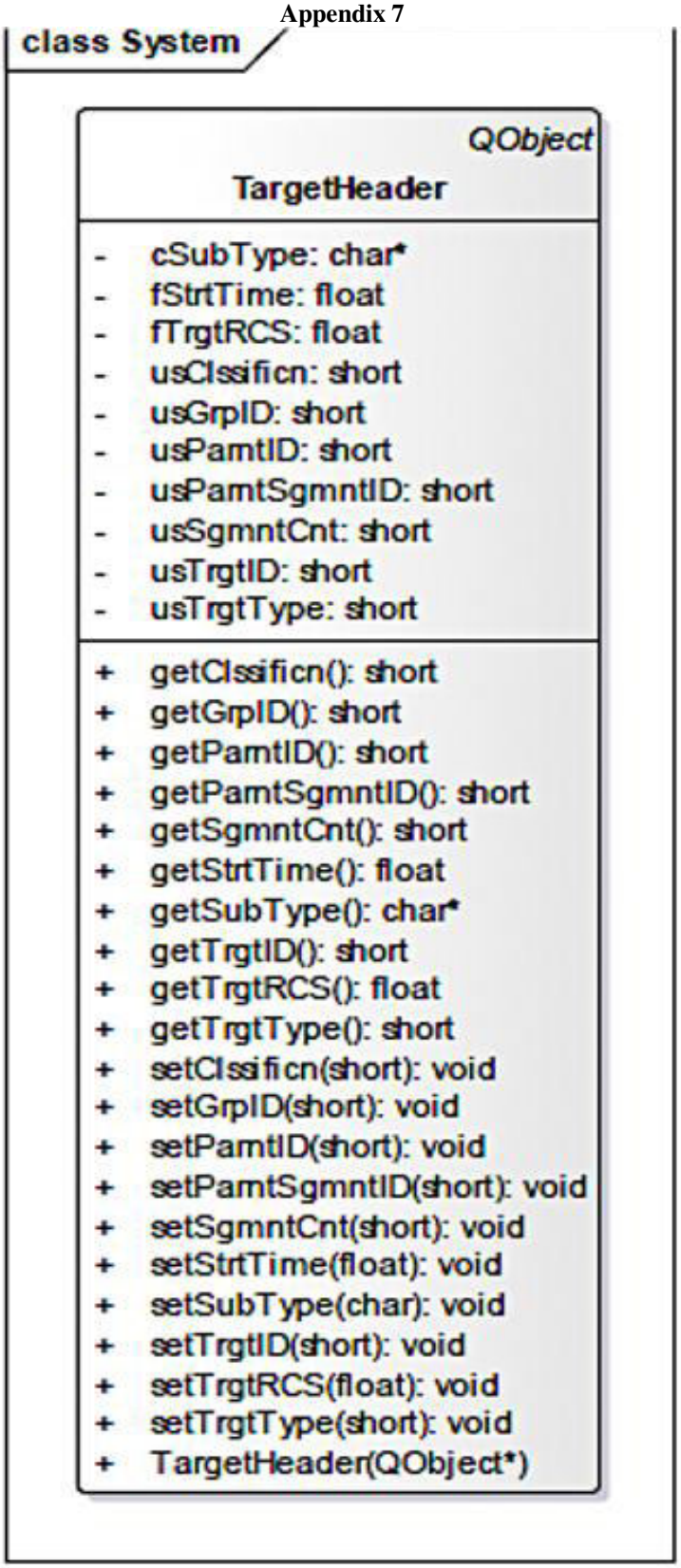
Appendix 6



Class 6: TargetPosition Class

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

Appendix 7



Class 7: TargetHeader Class

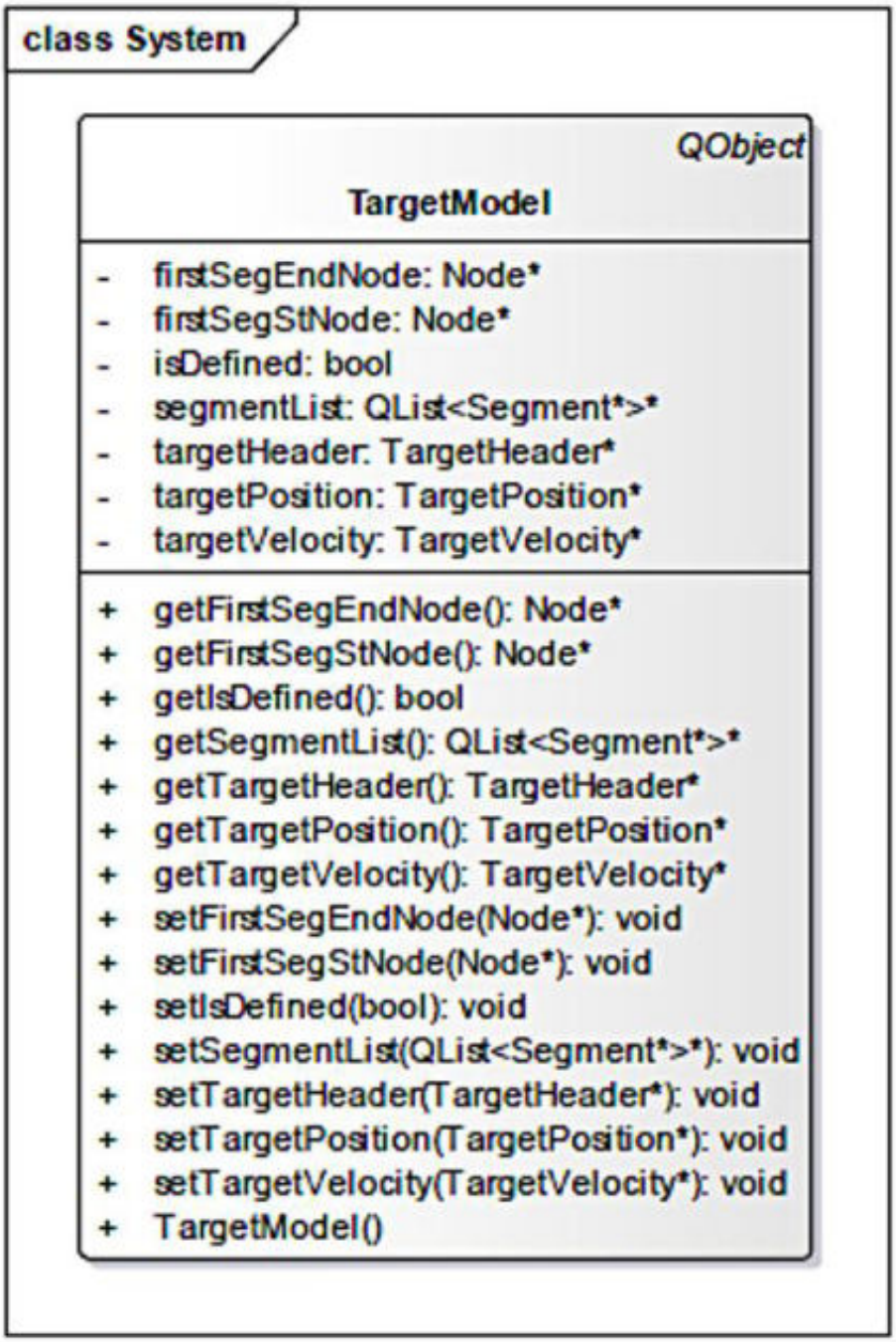
ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

Appendix 8

Class 8: ArcSegment Class

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

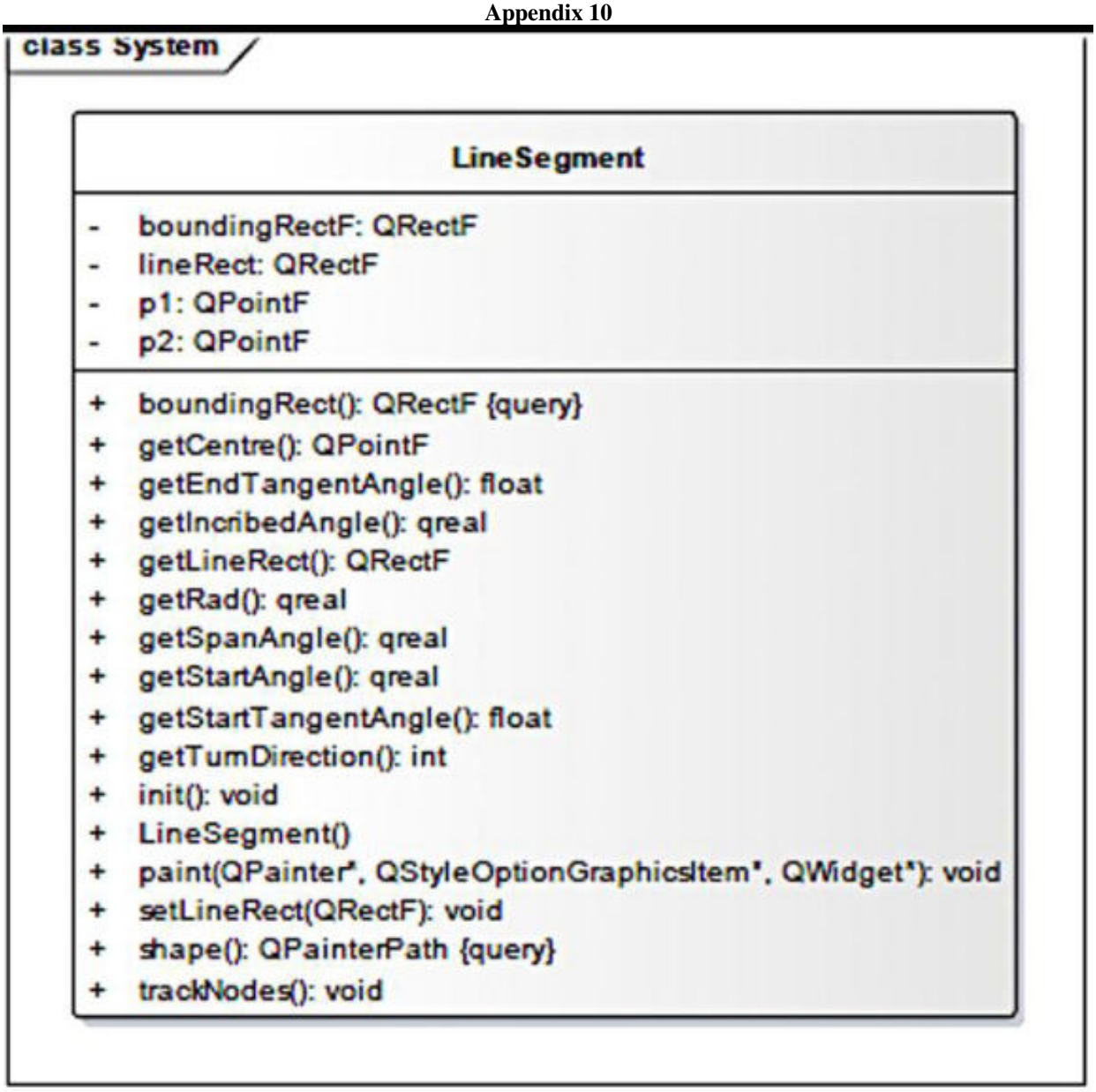
Appendix 9



Class 9: TargetModel Class

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

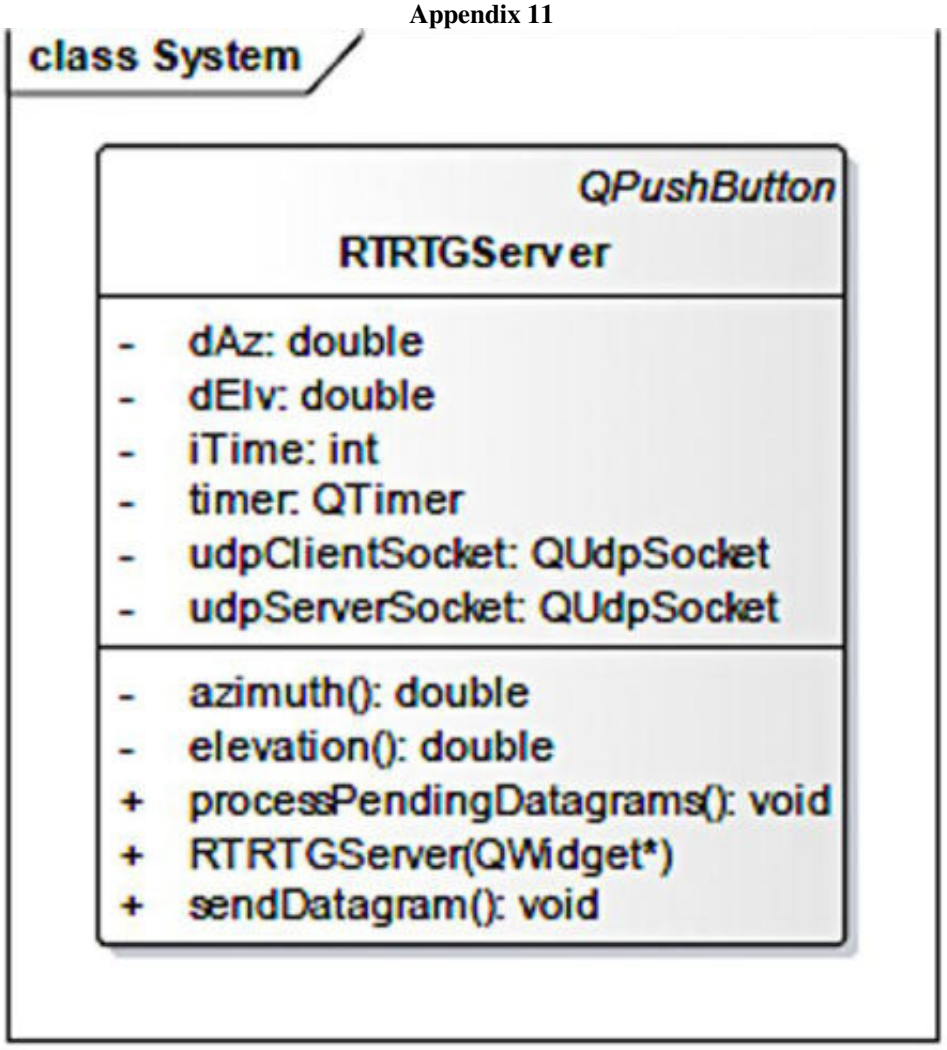
Appendix 10



Class 10: LineSegment Class

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

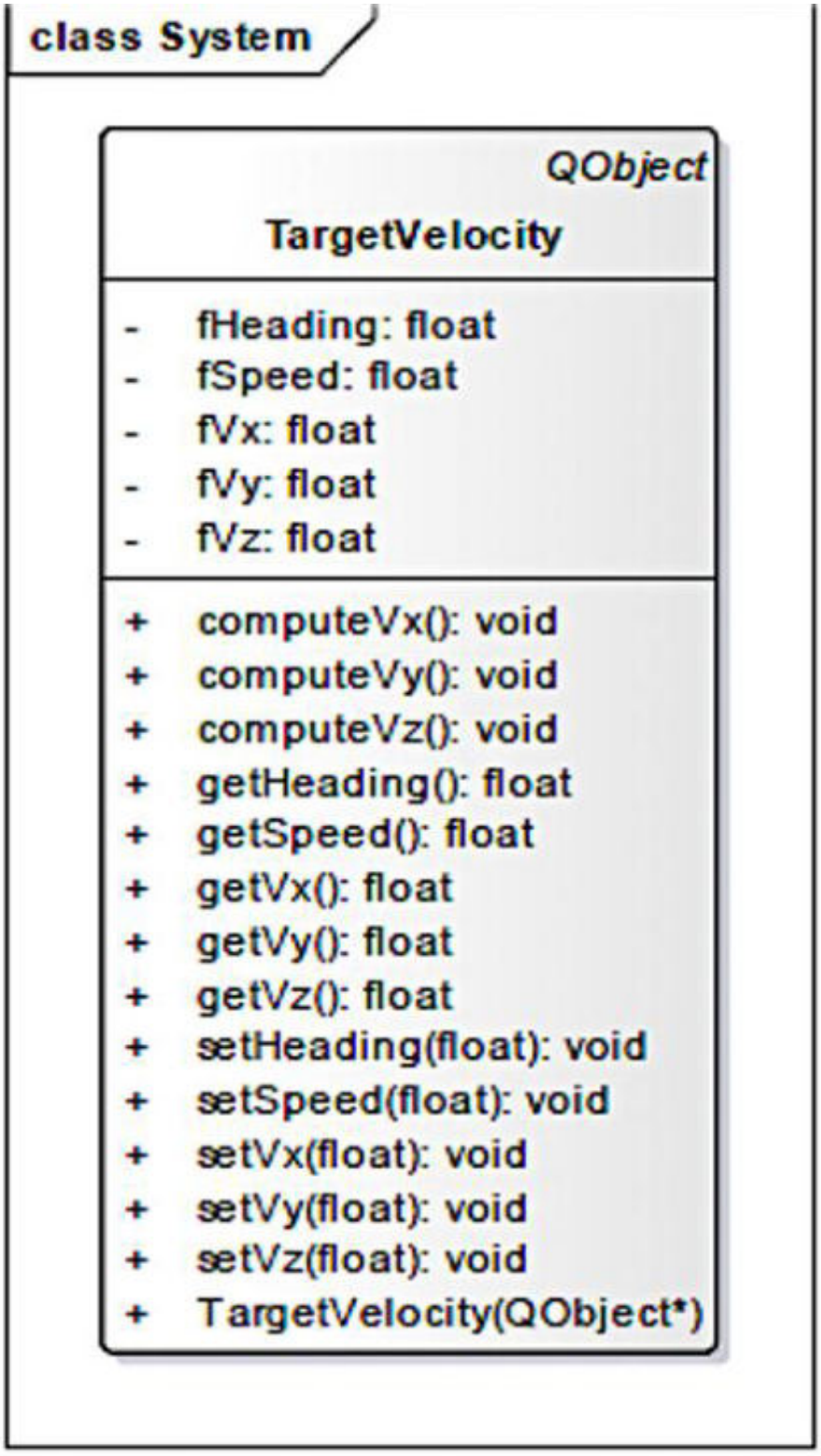
Appendix 11



Class 11: Server Class

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

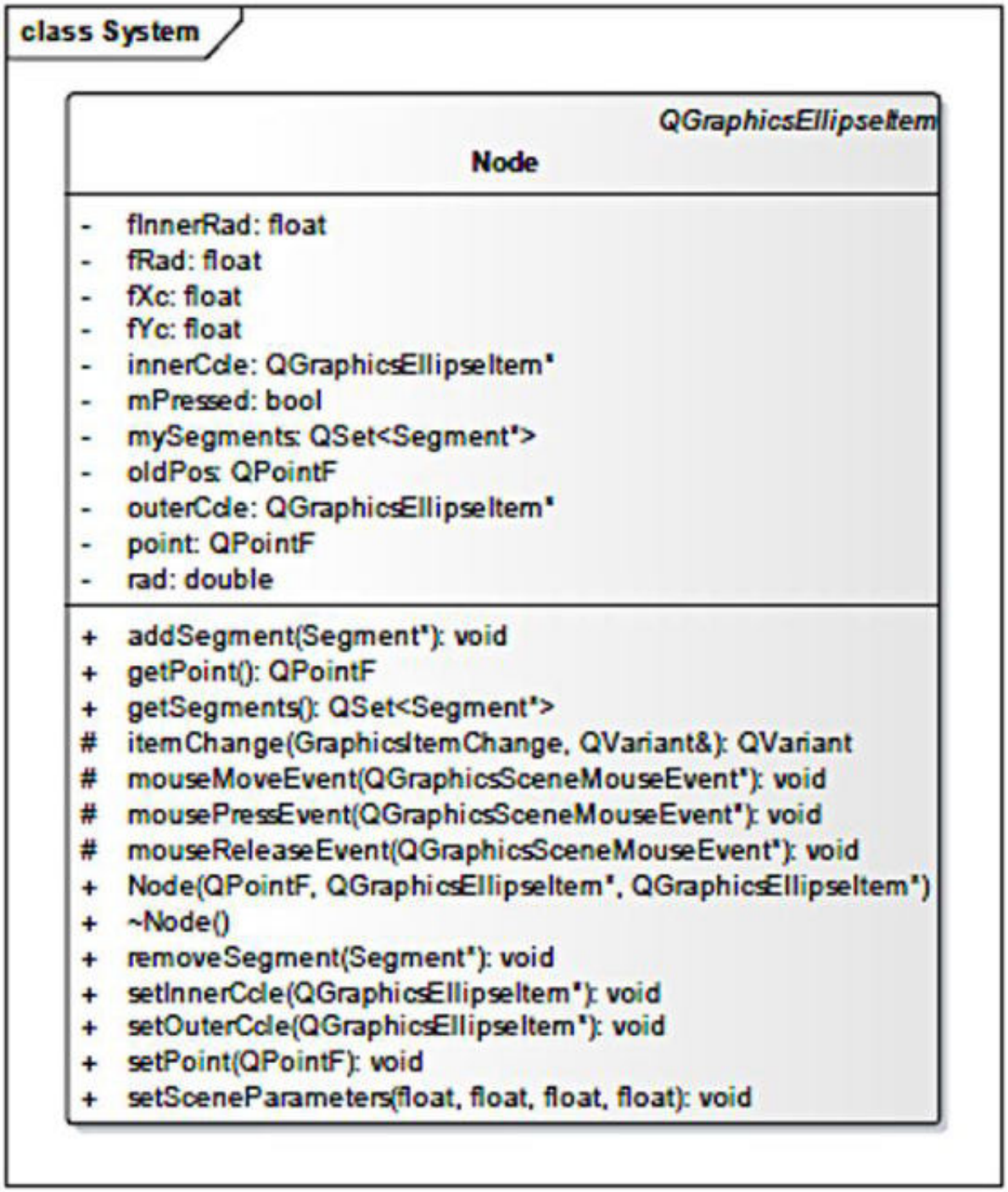
Appendix 12



Class 12: TargetVelocity Class

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

Appendix 13



Class 13: Node Class

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

Appendix 14

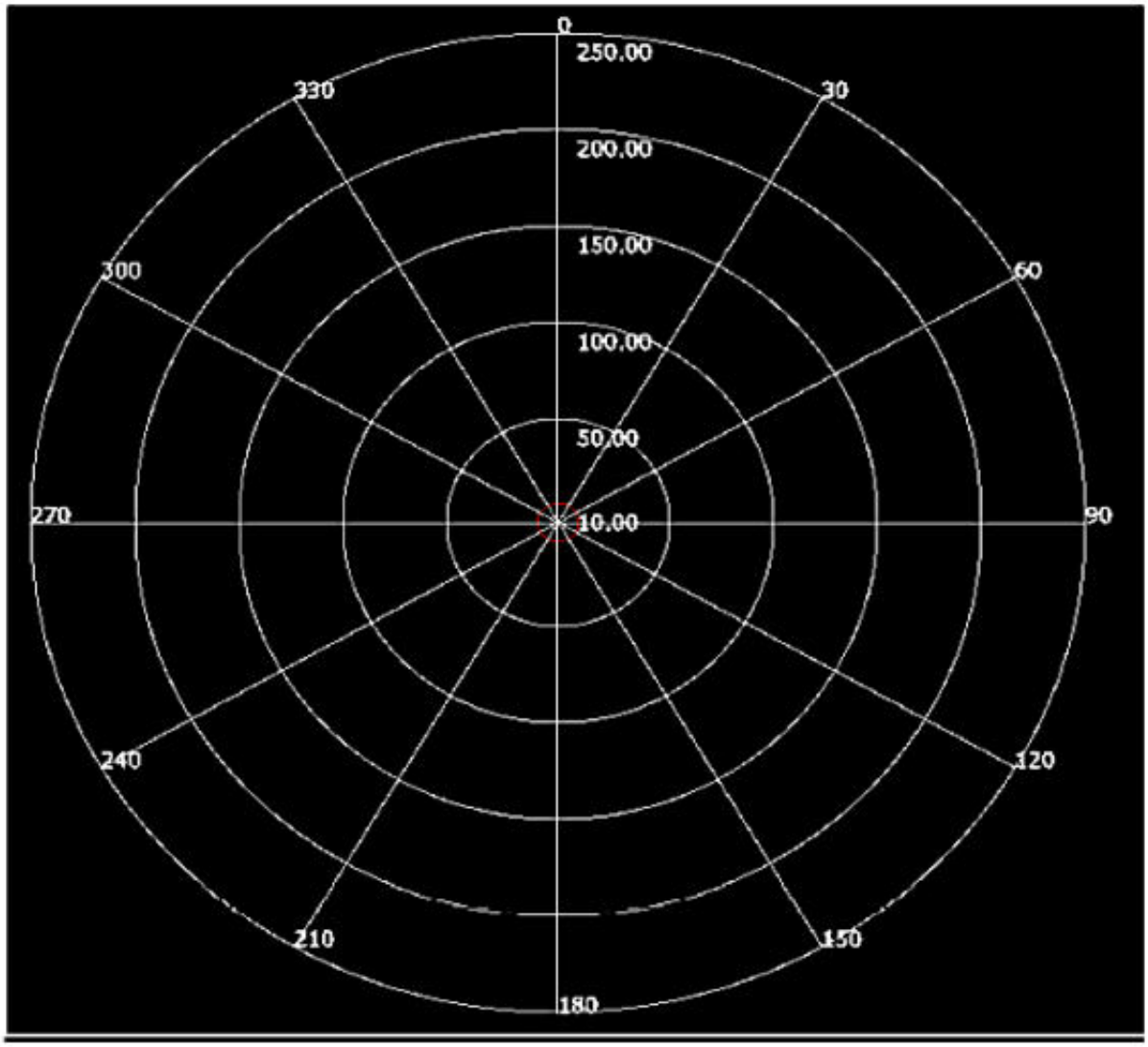


Figure 17: Radar PPI Screen

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

Appendix 15

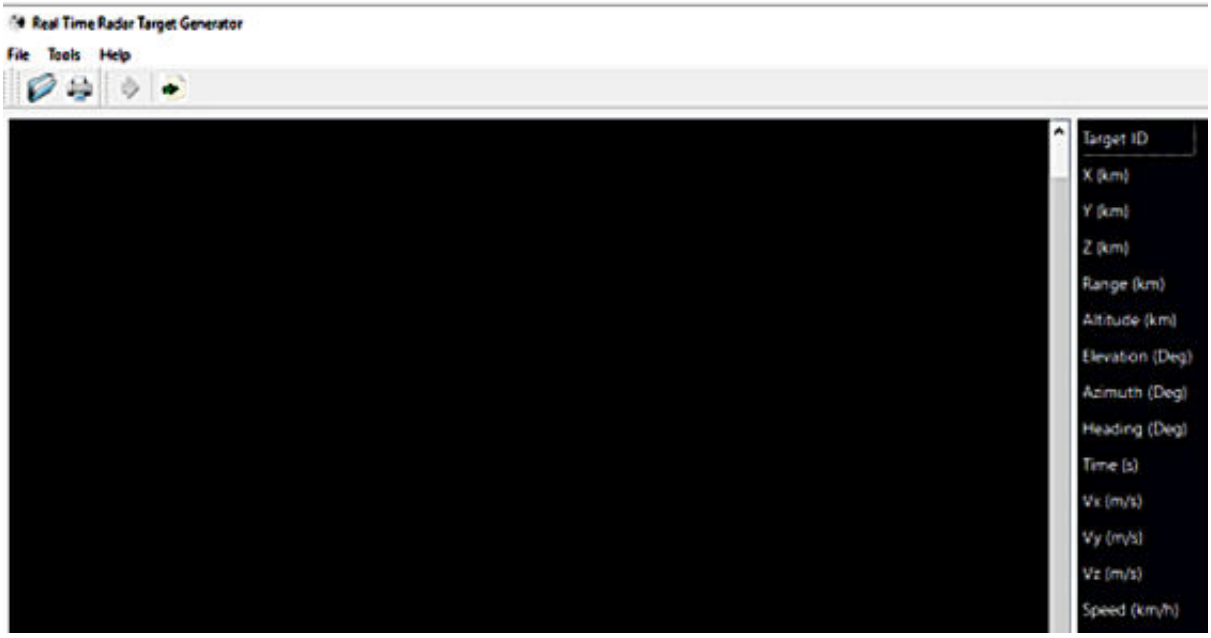


Figure 18: Main Window Screen

Appendix 15

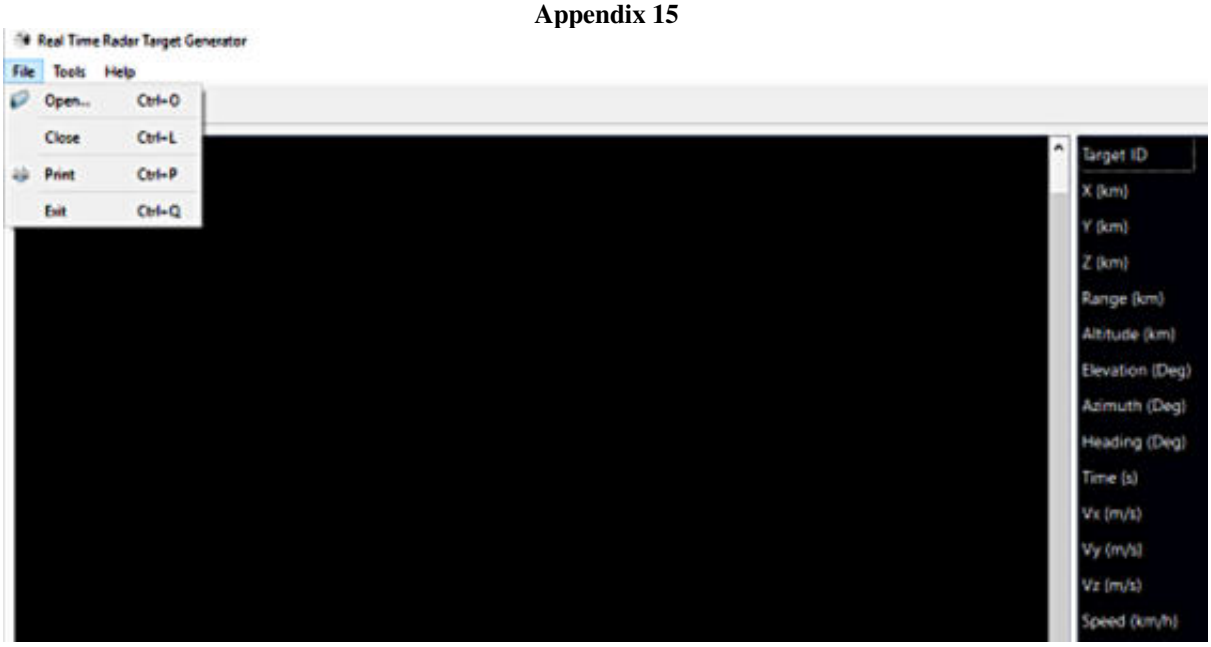


Figure 19: File Menu Screen

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

Appendix 16

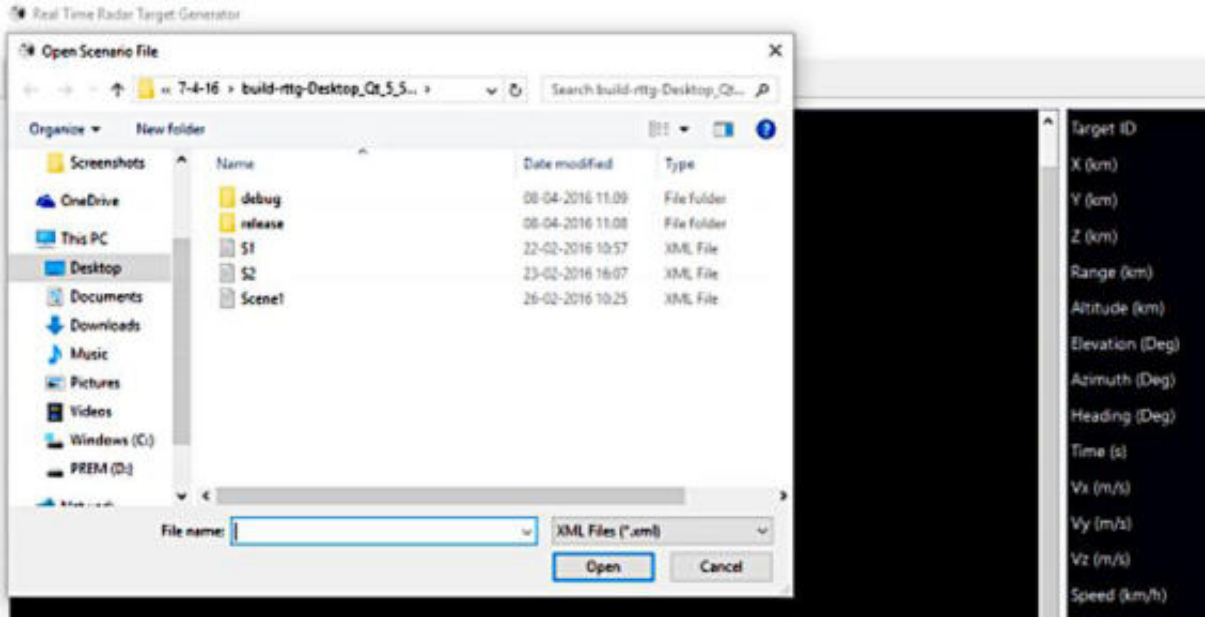


Figure 20: Open Scenario Screen

Appendix 17

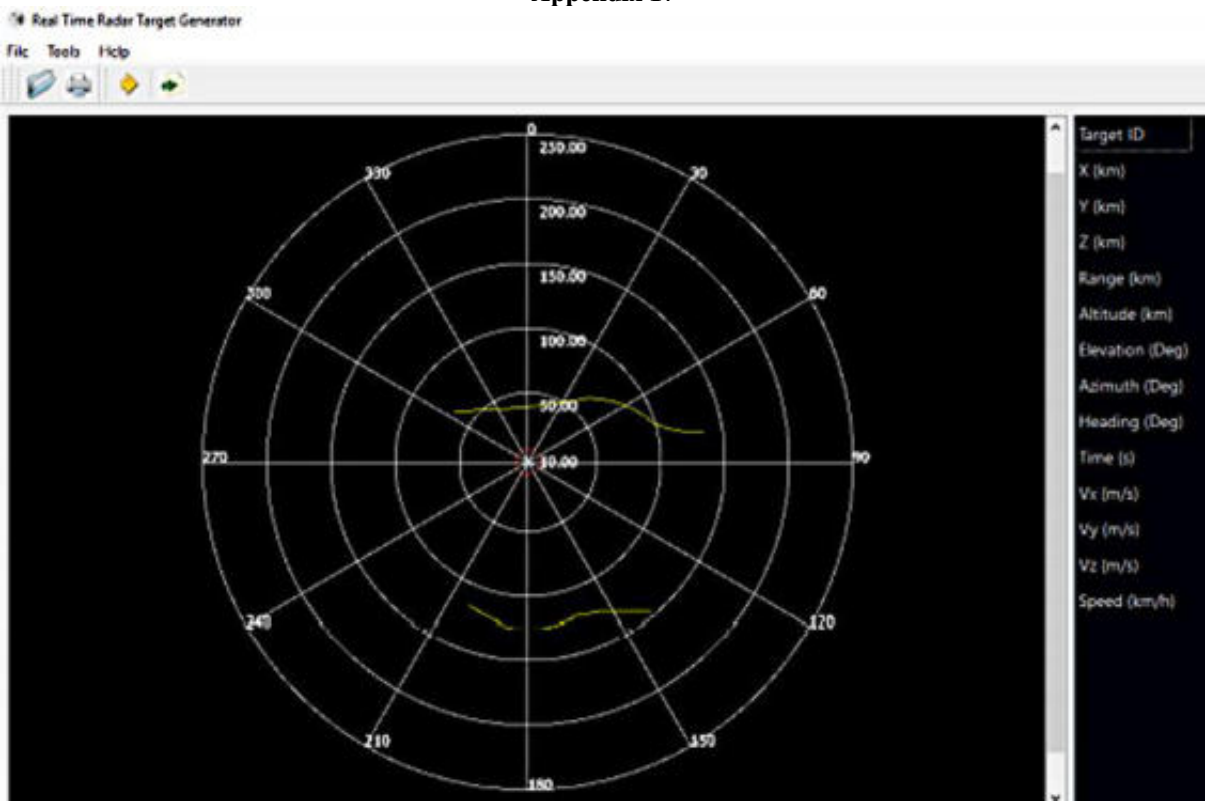


Figure 21: Scenario Display Screen

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

Appendix 18

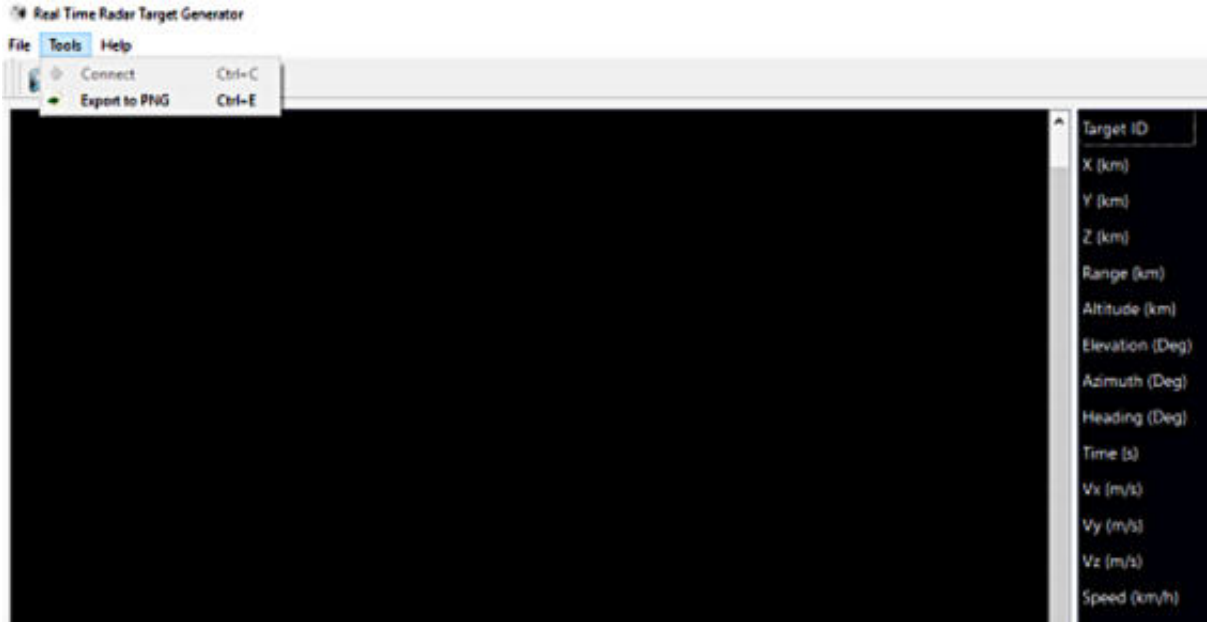


Figure 22: Tools Menu Screen

Appendix 19

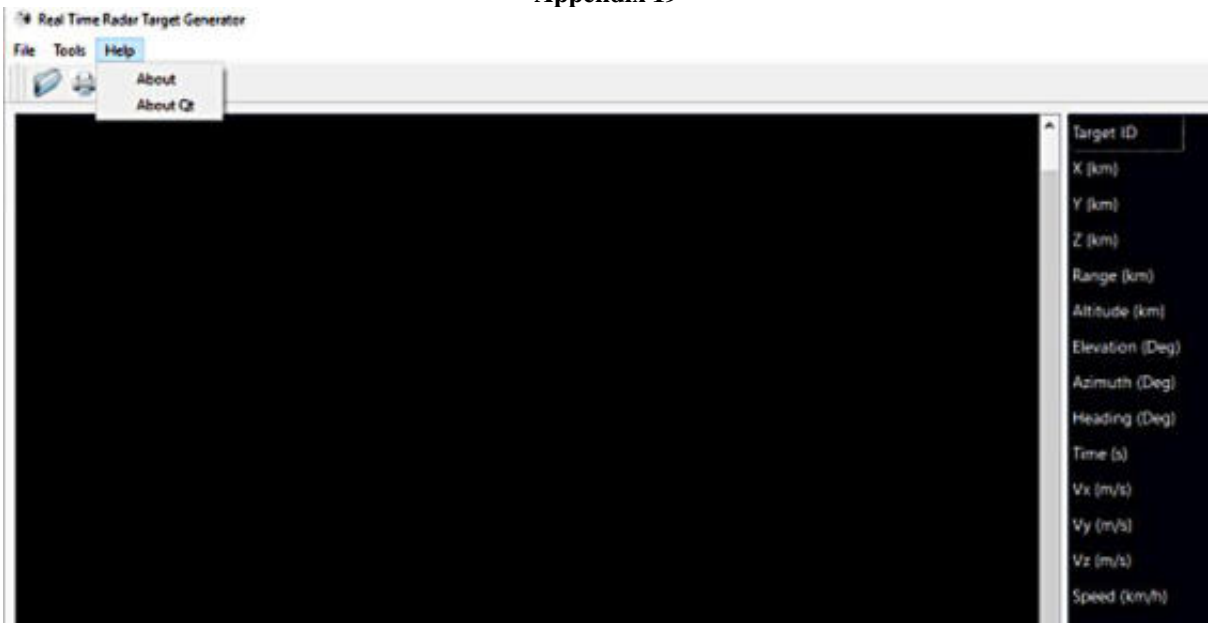


Figure 23: Help Menu Screen

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

Appendix 20

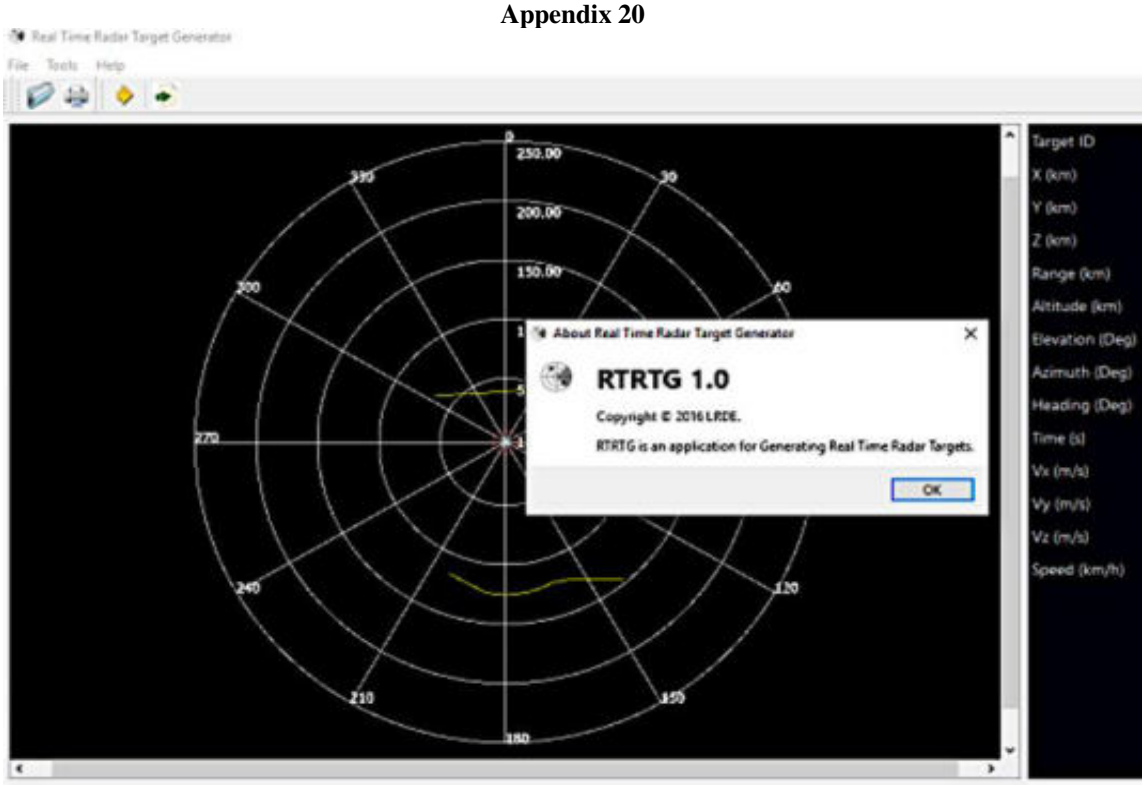


Figure 24 Version Menu Screen

Appendix 21

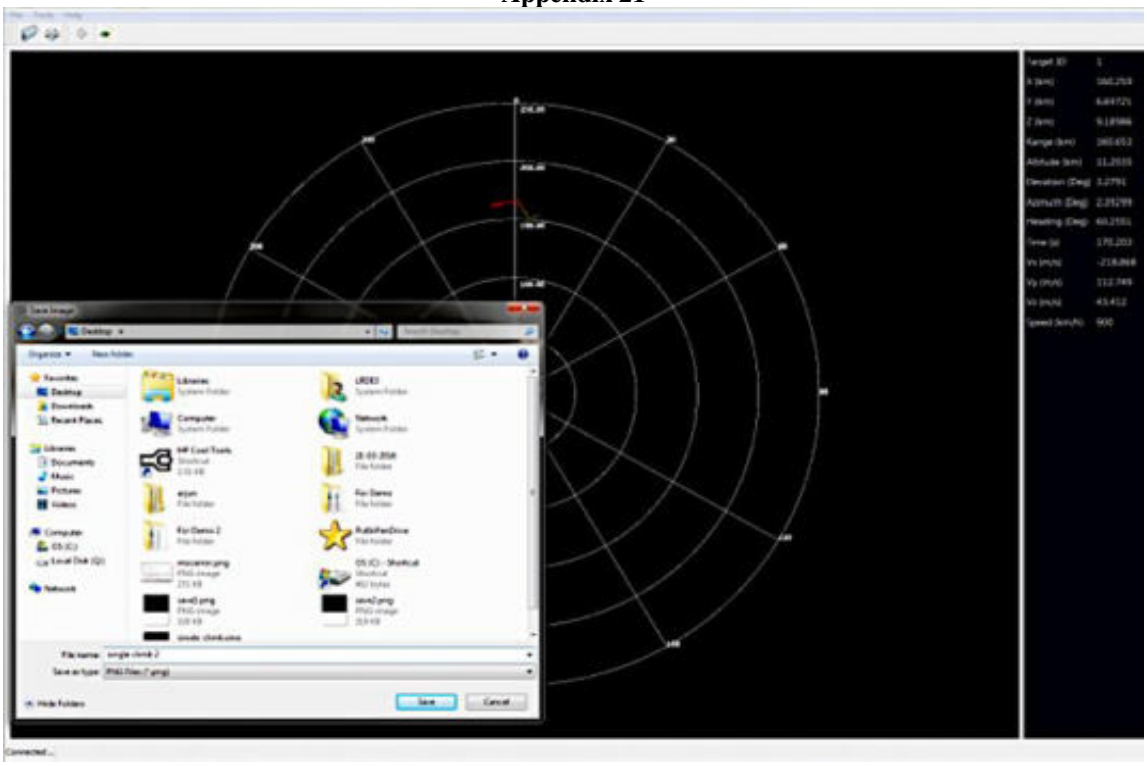


Figure 25: Save Image / Export to PNG Menu

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

Appendix 22

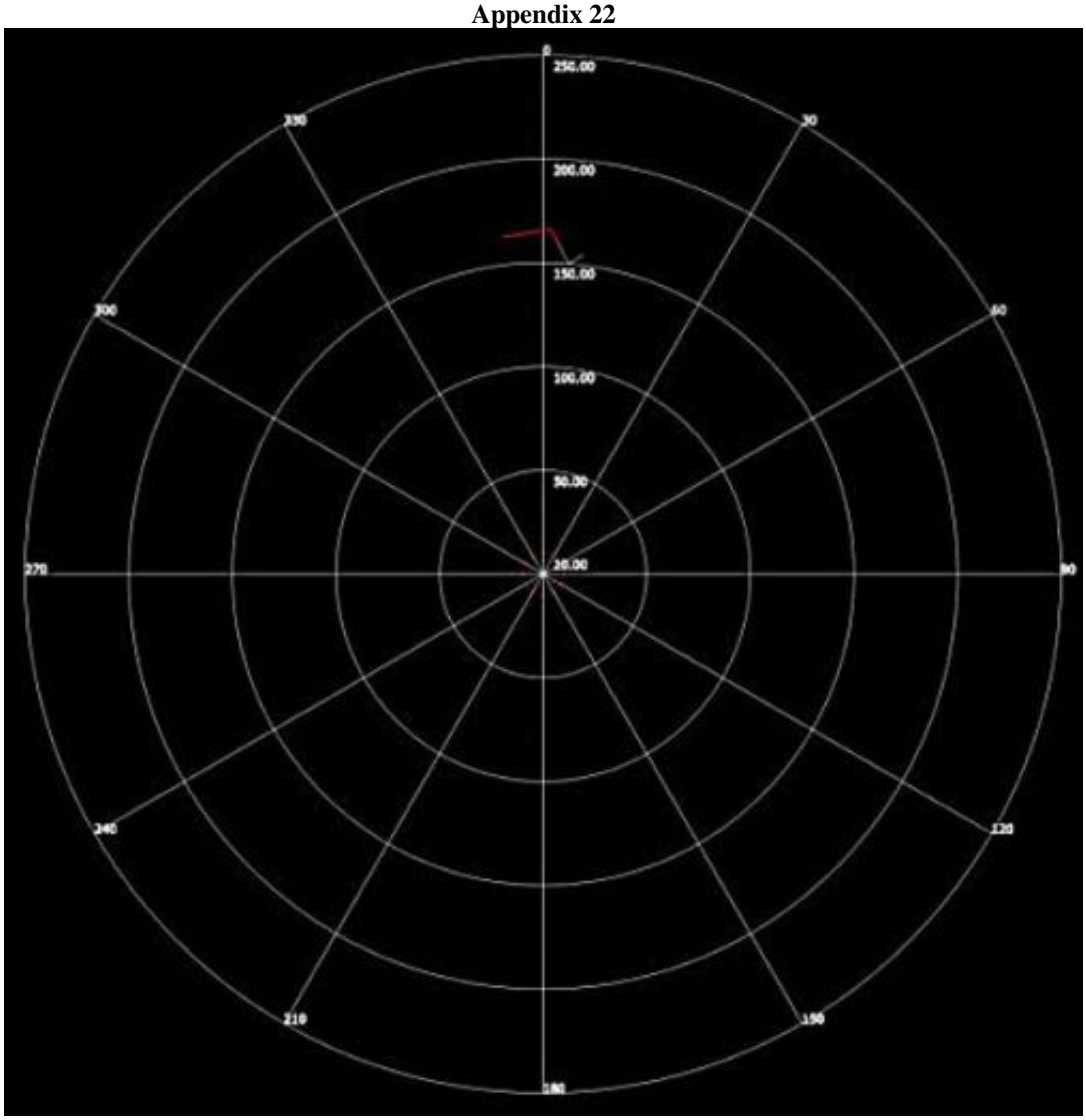


Figure 26: The Saved PNG Image From Appendix 21

ISSN (Online): 2583-2468
Volume-3 Issue-2 || March 2024 || PP. 1-34 DOI: 10.5281/zenodo.10846637

Appendix 23

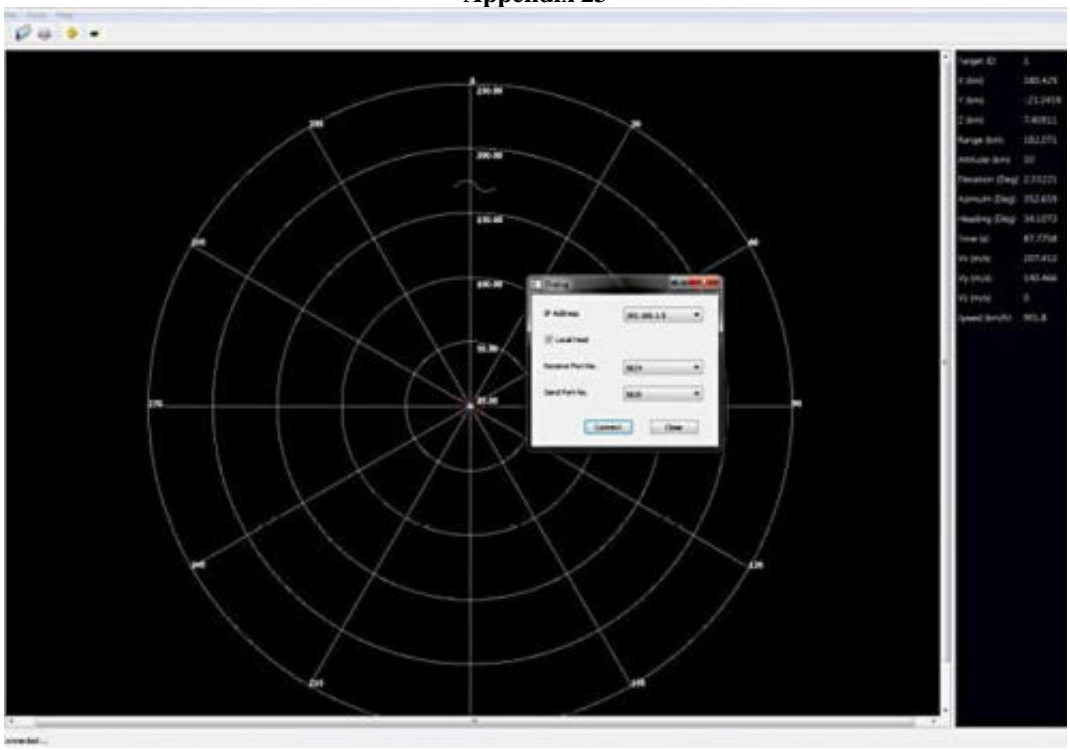


Figure 27: Connect Menu Screen

Appendix 24

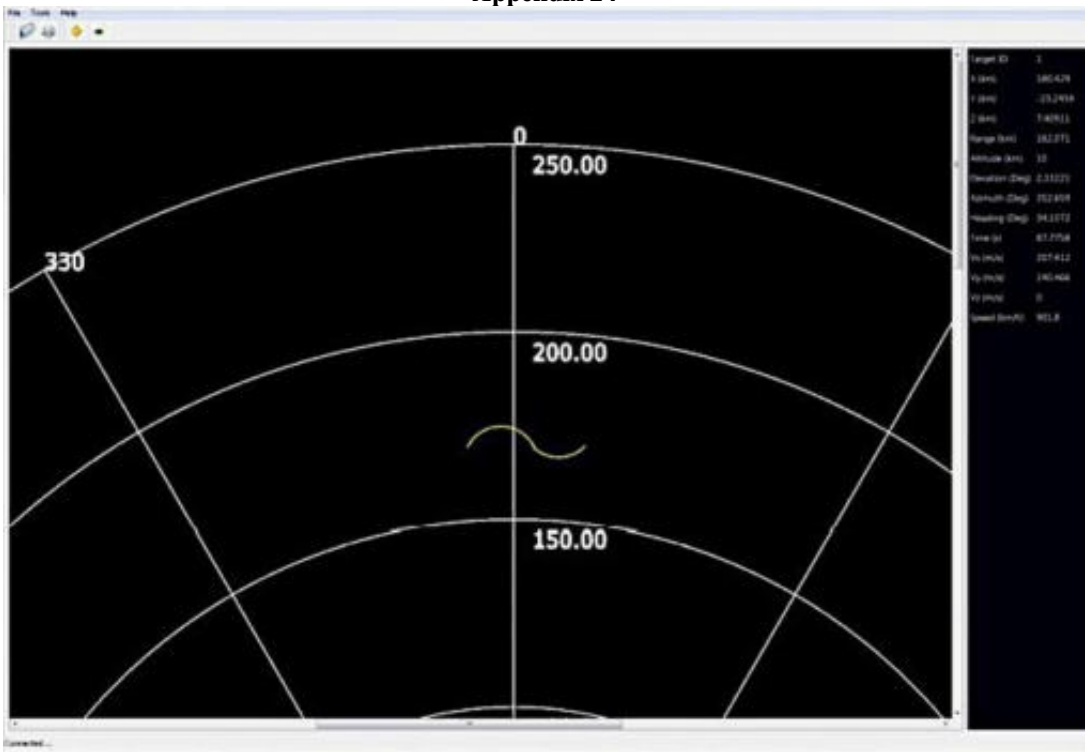


Figure 28: Zoomed Screen