



**Calhoun: The NPS Institutional Archive**  
**DSpace Repository**

---

Theses and Dissertations

1. Thesis and Dissertation Collection, all items

---

2008-03

**Modeling airport ground operations using  
Discrete Event Simulation (DES) and X3D visualization**

Ouerghi, Nabil

Monterey, California. Naval Postgraduate School

---

<http://hdl.handle.net/10945/4197>

*Downloaded from NPS Archive: Calhoun*



Calhoun is a project of the Dudley Knox Library at NPS, furthering the precepts and goals of open government and government transparency. All information contained herein has been approved for release by the NPS Public Affairs Officer.

**Dudley Knox Library / Naval Postgraduate School**  
**411 Dyer Road / 1 University Circle**  
**Monterey, California USA 93943**

<http://www.nps.edu/library>



**NAVAL  
POSTGRADUATE  
SCHOOL**

**MONTEREY, CALIFORNIA**

**THESIS**

**MODELING AIRPORT GROUND OPERATIONS  
USING DISCRETE EVENT SIMULATION (DES)  
AND X3D VISUALIZATION**

by

Nabil Ouerghi

March 2008

Thesis Advisor:  
Thesis Co-Advisors:

Don Brutzman  
Arnold Buss  
Terry Norbraten

**Approved for public release; distribution is unlimited**

THIS PAGE INTENTIONALLY LEFT BLANK

| <b>REPORT DOCUMENTATION PAGE</b>   |   |  | <i>Form Approved OMB No. 0704-0188</i>  |
|--|---|--|---|
| Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.  |   |  |   |
| <b>1. AGENCY USE ONLY (Leave blank)</b>  | <b>2. REPORT DATE</b><br>March 2008                             | <b>3. REPORT TYPE AND DATES COVERED</b><br>Master's Thesis     |   |
| <b>4. TITLE AND SUBTITLE</b> Modeling Airport Ground Operations using Discrete Event Simulation (DES) and X3d Visualization  |   | <b>5. FUNDING NUMBERS</b>                                      |   |
| <b>6. AUTHOR</b> Nabil Ouerghi   |   |  |   |
| <b>7. PERFORMING ORGANIZATION NAME AND ADDRESS</b><br>Naval Postgraduate School<br>Monterey, CA 93943-5000   |   | <b>8. PERFORMING ORGANIZATION REPORT NUMBER</b>                |   |
| <b>9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b><br>N/A   |   | <b>10. SPONSORING/MONITORING AGENCY REPORT NUMBER</b>          |   |
| <b>11. SUPPLEMENTARY NOTES</b> The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.  |   |  |   |
| <b>12a. DISTRIBUTION / AVAILABILITY STATEMENT</b><br>Approved for public release; distribution is unlimited  |   | <b>12b. DISTRIBUTION CODE</b>                                  |   |
| <b>13. ABSTRACT</b><br><br><p>Almost all flight simulators are centered on the problems that can occur during flight, whereas airport ground traffic problems are seldom addressed and are growing considerably. A number of precautions have been directed by the U.S. Federal Aviation Administration (FAA) to overcome these challenges, such as pilot training and adding taxiway indicator signals to help pilots follow specific paths when taxiing. Further work is needed.</p> <p>This thesis simulates the problem of Ground Traffic incursions. Discrete Event Simulation (DES) and the Viskit tool are used to build two scenarios describing the takeoff and the landing maneuvers including potential ground incidents. It also presents the different techniques used to build 3D graphics models for the airplanes and the airport environment using Extensible 3D (X3D) graphics. After running the simulation a number of times with different parameters, collected data support basic analysis and potential conclusions. This approach demonstrates a proof-of-concept capability worthy of future work.</p> |   |  |   |
| <b>14. SUBJECT TERMS</b><br>Discrete Event Simulation, DES, Simkit, Diskit, Viskit, Savage, XML, Distributed Interactive Simulation, DIS, Blender, X3D Edit  |   | <b>15. NUMBER OF PAGES</b><br>113                              |   |
|  |   | <b>16. PRICE CODE</b>  |   |
| <b>17. SECURITY CLASSIFICATION OF REPORT</b><br>Unclassified   | <b>18. SECURITY CLASSIFICATION OF THIS PAGE</b><br>Unclassified | <b>19. SECURITY CLASSIFICATION OF ABSTRACT</b><br>Unclassified | <b>20. LIMITATION OF ABSTRACT</b><br>UU |

THIS PAGE INTENTIONALLY LEFT BLANK

**Approved for public release; distribution is unlimited**

**MODELING AIRPORT GROUND OPERATIONS  
USING DISCRETE EVENT SIMULATION (DES)  
AND X3D VISUALIZATION**

Nabil Ouerghi  
First Lieutenant, Tunisia Air Force  
Academy of The Air Force Borj El Amri, 2001

Submitted in partial fulfillment of the  
requirements for the degree of

**MASTER OF SCIENCE IN MODELING, VIRTUAL ENVIRONMENTS  
AND SIMULATION (MOVES)**

from the

**NAVAL POSTGRADUATE SCHOOL  
March 2008**

Author: Nabil Ouerghi

Approved by: Don Brutzman  
Thesis Advisor

Arnold Buss  
Co-Advisor

Terry Norbraten  
Co-Advisor

Mathias Kolsch  
Chair, MOVES Academic Committee

THIS PAGE INTENTIONALLY LEFT BLANK

## **ABSTRACT**

Almost all flight simulators are centered on the problems that can occur during flight, whereas airport ground traffic problems are seldom addressed and are growing considerably. A number of precautions have been directed by the U.S. Federal Aviation Administration (FAA) to overcome these challenges, such as pilot training and adding taxiway indicator signals to help pilots follow specific paths when taxiing. Further work is needed.

This thesis simulates the problem of Ground Traffic incursions. Discrete Event Simulation (DES) and the Viskit tool are used to build two scenarios describing the takeoff and the landing maneuvers including potential ground incidents. It also presents the different techniques used to build 3D graphics models for the airplanes and the airport environment using Extensible 3D (X3D) graphics. After running the simulation a number of times with different parameters, collected data support basic analysis and potential conclusions. This approach demonstrates a proof-of-concept capability worthy of future work.



THIS PAGE INTENTIONALLY LEFT BLANK

# TABLE OF CONTENTS

|             |   |           |
|-------------|---|-----------|
| <b>I.</b>   | <b>INTRODUCTION.....</b>  | <b>1</b>  |
| <b>A.</b>   | <b>OVERVIEW .....</b>   | <b>1</b>  |
| <b>B.</b>   | <b>PROBLEM STATEMENT .....</b>  | <b>2</b>  |
| <b>C.</b>   | <b>MOTIVATION .....</b>   | <b>2</b>  |
| <b>D.</b>   | <b>OBJECTIVES .....</b>   | <b>3</b>  |
| <b>E.</b>   | <b>THESIS ORGANIZATION.....</b>   | <b>4</b>  |
| <b>II.</b>  | <b>BACKGROUND AND RELATED WORK .....</b>  | <b>5</b>  |
| <b>A.</b>   | <b>INTRODUCTION.....</b>  | <b>5</b>  |
| <b>B.</b>   | <b>DISCRETE EVENT SIMULATION (DES).....</b>                                       | <b>5</b>  |
| <b>1.</b>   | <b>Event Graph .....</b>  | <b>5</b>  |
| <b>2.</b>   | <b>Arrival Process Example.....</b>   | <b>6</b>  |
| <b>C.</b>   | <b>ROLE OF 3D VISUALIZATION IN DISCRETE EVENT<br/>SIMULATION .....</b>            | <b>7</b>  |
| <b>D.</b>   | <b>X3D GRAPHICS.....</b>  | <b>7</b>  |
| <b>1.</b>   | <b>X3D-Edit Authoring Tool .....</b>  | <b>8</b>  |
| <b>2.</b>   | <b>XJ3D Open Source Project For X3D .....</b>                                     | <b>9</b>  |
| <b>3.</b>   | <b>VIZX3D / FLUX Studio Scene Authoring Tool .....</b>                            | <b>9</b>  |
| <b>4.</b>   | <b>WINGS3D Authoring Tool .....</b>   | <b>10</b> |
| <b>5.</b>   | <b>Blender Authoring Tool .....</b>   | <b>11</b> |
| <b>E.</b>   | <b>IEEE DISTRIBUTED INTERACTIVE SIMULATION (DIS)<br/>NETWORK PROTOCOL .....</b>   | <b>12</b> |
| <b>F.</b>   | <b>SAVAGE MODELING ANALYSIS LANGUAGE (SMAL) FOR<br/>SIMULATION METADATA .....</b> | <b>13</b> |
| <b>G.</b>   | <b>SUMMARY .....</b>  | <b>14</b> |
| <b>III.</b> | <b>BEHAVIOR MODELING CAPABILITIES .....</b>                                       | <b>15</b> |
| <b>A.</b>   | <b>INTRODUCTION.....</b>  | <b>15</b> |
| <b>B.</b>   | <b>DISCRETE EVENT PROGRAMMING WITH SIMKIT .....</b>                               | <b>15</b> |
| <b>1.</b>   | <b>SimEntity and SimEntityBase .....</b>  | <b>15</b> |
| <b>2.</b>   | <b>Simple Movement and Detection in DES.....</b>                                  | <b>16</b> |
| <b>3.</b>   | <b>Mover3D and DISMover3D .....</b>   | <b>18</b> |
| <b>C.</b>   | <b>EVENT GRAPH EDITOR IN VISKIT .....</b>   | <b>18</b> |
| <b>1.</b>   | <b>Event Graph Parameters .....</b>   | <b>19</b> |
| <b>2.</b>   | <b>Defining Event Parameters .....</b>  | <b>21</b> |
| <b>D.</b>   | <b>ASSEMBLY EDITOR IN VISKIT .....</b>  | <b>21</b> |
| <b>1.</b>   | <b>Creating a Simple Assembly .....</b>   | <b>21</b> |
| <b>2.</b>   | <b>Property Change Listener (PCL) .....</b>                                       | <b>24</b> |
| <b>3.</b>   | <b>The Assembly Run .....</b>   | <b>25</b> |
| <b>4.</b>   | <b>The Assembly Report .....</b>  | <b>26</b> |
| <b>E.</b>   | <b>SUMMARY .....</b>  | <b>26</b> |
| <b>IV.</b>  | <b>MODEL IMPLEMENTATION USING VISKIT.....</b>                                     | <b>29</b> |

|      |   |    |
|------|---|----|
| A.   | INTRODUCTION.....   | 29 |
| B.   | METHODOLOGY .....   | 29 |
| C.   | USE CASES.....  | 29 |
| D.   | SCENARIO NARRATIVES.....                                    | 30 |
| E.   | RIGHT OF WAY RULES .....                                    | 31 |
| F.   | ENTITIES AND DATA STRUCTURES.....                           | 32 |
| G.   | EVENT GRAPH MODELS FOR THE TWO SCENARIOS.....               | 34 |
|      | 1. First Scenario .....                                     | 34 |
|      | 2. Final Scenario.....                                      | 37 |
| H.   | ASSEMBLY MODELS FOR FIRST AND FINAL SCENARIOS .....         | 40 |
|      | 1. First Scenario .....                                     | 40 |
|      | 2. Final Scenario.....                                      | 42 |
| I.   | MEASURE OF EFFECTIVENESS (MOES) FOR BOTH<br>SCENARIOS ..... | 44 |
| J.   | SUMMARY .....   | 44 |
| V.   | VISUALIZATION OF THE GROUND TRAFFIC WITH X3D GRAPHICS ....  | 45 |
| A.   | INTRODUCTION.....   | 45 |
|      | 1. Data Collection .....                                    | 45 |
|      | a. Imagery Data.....  | 46 |
|      | b. Elevation Data.....                                      | 46 |
|      | c. Terrain Generation .....                                 | 49 |
|      | d. Rez Tool.....  | 49 |
|      | 2. Building the Airplane .....                              | 51 |
|      | a. Meta Tags .....  | 54 |
|      | b. Scene Corrections .....                                  | 54 |
|      | c. IndexFaceSet Corrections .....                           | 55 |
|      | 3. Model Integration into Master Scene.....                 | 56 |
| B.   | SUMMARY .....   | 58 |
| VI.  | SIMULATION ANALYSIS .....                                   | 59 |
| A.   | INTRODUCTION.....   | 59 |
| B.   | STATISTICS AND ANALYST REPORT .....                         | 59 |
|      | 1. Takeoff Scenario .....                                   | 59 |
|      | 2. Final Scenario.....                                      | 60 |
|      | 3. Design of Experiment and Data Collection .....           | 60 |
| C.   | SUMMARY .....   | 63 |
| VII. | CONCLUSIONS AND RECOMMENDATIONS.....                        | 65 |
| A.   | CONCLUSIONS .....   | 65 |
|      | 1. Building the Event Graph .....                           | 65 |
|      | 2. Implementing the Assembly.....                           | 65 |
|      | 3. Building the 3D Environment and Models.....              | 65 |
| B.   | RECOMMENDATIONS FOR FUTURE WORK.....                        | 66 |
|      | 1. Scenario Improvement .....                               | 66 |
|      | 2. Modeling Constraints .....                               | 67 |
|      | 3. Implementing the Model in SavageStudio:.....             | 67 |

**APPENDIX A. ANALYST REPORT FOR THE TAKEOFF SCENARIO .....71**  
**APPENDIX B. ANALYST REPORT FOR THE FINAL SCENARIO  
TAKEOFF AND LANDING PROCEDURES .....79**  
**LIST OF REFERENCES .....89**  
**INITIAL DISTRIBUTION LIST .....91**

THIS PAGE INTENTIONALLY LEFT BLANK

## LIST OF FIGURES

|            |  |    |
|------------|--|----|
| Figure 1.  | Fundamental event graph representation .....   | 6  |
| Figure 2.  | Simple event graph representation (Buss 2001) .....  | 7  |
| Figure 3.  | A snapshot of the X3D-Edit user interface for authoring X3D models showing a model of airplane via XJ3D browser.....   | 8  |
| Figure 4.  | A snapshot of the Xj3D browser showing a model of terrain in X3D format.....   | 9  |
| Figure 5.  | Flux Studio user interface for authoring X3D models.....   | 10 |
| Figure 6.  | Wings 3D User Interface for creating 3D models that can be exported to X3D format. ....  | 11 |
| Figure 7.  | The Blender interface showing different views of an airplane construction. ...   | 12 |
| Figure 8.  | The Distributed Interactive Simulation (DIS) family of standards.....  | 13 |
| Figure11.  | The code associated to the state transition for event A in the basic event graph example.....  | 16 |
| Figure 12. | The moving entity detected by the sensor just after it enters the sensor range and undetected just before exiting the sensor range(Buss and Sanchez 2005)..... | 17 |
| Figure 13. | Illustration of the event graph showing the different components of the DisMover3D (Sullivan 2006). ....   | 18 |
| Figure 14. | A snapshot of Viskit tool showing the user interface with the different tools.....   | 19 |
| Figure 15. | Snapshot depicting the procedure of creating a new event graph with Viskit application.....  | 20 |
| Figure 16. | The menu in which the author can parameterize the event.....   | 21 |
| Figure 17. | Assembly editor in the Viskit tool where the scenario author can use an already-created event graph. ....  | 22 |
| Figure 18. | The menu allowing the author to configure the new created assembly.....  | 22 |
| Figure 19. | The Manipulation of the saved event graph models in the assembly editor menu.....  | 23 |
| Figure 20. | The different connectors in the Assembly Editor with their different properties.....   | 23 |
| Figure 21. | Different Property Change Listeners (PCL) in the menu of Assembly Editor.....  | 24 |
| Figure 22. | The interface where the author can configure and run the simulation.....   | 25 |
| Figure 23. | Summary of Tables generated by the Analysis Report after the simulation stops running.....   | 26 |
| Figure 24. | The different stationary points in the Environment of the simulation for the takeoff scenario.....   | 31 |
| Figure 25. | The event graph of the Takeoff scenario showing the different stages the airplanes go through before taking off.....   | 36 |
| Figure 26. | The final scenario implementation in event graph model.....  | 39 |
| Figure 27. | Simple Arrival Process event graph responsible for incrementing the arrival of airplanes. ....   | 40 |

|            |   |    |
|------------|---|----|
| Figure 28. | Creator event graph responsible for creating airplane entities.....   | 41 |
| Figure 29. | Assembly of the Takeoff scenario with a simple property dumper to track the different linked lists containing the airplanes. ....                                       | 41 |
| Figure 30. | Assembly of the final scenario with two property change listeners. ....   | 43 |
| Figure 31. | Available online data sources for imagery and elevation using Global Mapper. ....   | 46 |
| Figure 32. | The USGS interface for downloading elevation data. ....   | 47 |
| Figure 33. | The interface the user may use to specify the region and observe the availability of the data.....  | 47 |
| Figure 34. | The menu allowing the download the elevation data.....  | 48 |
| Figure 35. | The user interface of Global Mapper used to download the airport elevation data. ....   | 49 |
| Figure 36. | The menu that appears when the Rez application is launched.....   | 50 |
| Figure 37. | The 3D terrain of Tunisia Carthage airport generated by Rez and displayed with XJ3D browser. ....   | 51 |
| Figure 38. | Blender interface with Different views of the Boeing 747. ....  | 52 |
| Figure 38. | Blender interface showing the model of the airplane with minimal vertices. ....   | 52 |
| Figure 40. | Blender interface with the export feature to X3D model.....   | 53 |
| Figure 41. | The four changes to make to the model after exporting from Blender to X3D edit.....   | 54 |
| Figure 42. | The model of Boeing 747 after making all the required changes on the exported model.....  | 56 |
| Figure 43. | The Tunisia Carthage airport X3D model in Savage Archive view with XJ3D browser. ....   | 57 |
| Figure 44. | An X3D scene combining the Boeing 747 model and the Tunisia Carthage airport model. Note proper scale modeling satellite imagery of aircraft next to the terminal. .... | 57 |
| Figure 45. | The process of generating the 17 combinations of the four factors with NOLH. ....   | 61 |
| Figure 46. | Distribution analysis of T1, T2, T3, and T4 as well as the average time spent by the three arriving airplane on the taxiway. ....                                       | 61 |
| Figure 47. | The concentration of the time spent on the taxiway for the first airplane, the second airplane and the third airplane.....  | 62 |
| Figure 48. | Blue circles showing some outliers for the average of time spent by the arriving airplanes on the taxiway.....  | 63 |
| Figure 49. | A snapshot of SavageStudio tool illustrating the airport model to the center and the airplane model to the upper right.....   | 68 |
| Figure 50. | A snapshot of the generated scene via SavageStudio and viewed with XJ3D browser, showing an airplane on the runway. ....  | 69 |

## LIST OF TABLES

|          |  |
|----------|--|
| Table 1. | The different connectors to the assemblies used for both scenarios .....24       |
| Table 2. | The different possible simultaneous events with their priorities .....32         |
| Table 3. | Different simulation components with their following data structure type .....34 |
| Table 4. | The different events composing the First Scenario.....36                         |
| Table 5. | The procedure of landing translated into event graph structure .....38           |
| Table 6. | Description of the assembly of the first scenario.....42                         |
| Table 7. | Different components forming the final scenario assembly. ....43                 |



THIS PAGE INTENTIONALLY LEFT BLANK

## LIST OF ACRONYMS AND ABBREVIATIONS

|      |   |
|------|---|
| API  | Application Programming Interface                 |
| ATO  | Air Tasking Order                                 |
| DIS  | Distributed Interactive Simulation                |
| DES  | Discrete Event Simulation                         |
| DOD  | Department of Defense                             |
| DOE  | Design of Experiments                             |
| FAA  | Federal Aviation Administration                   |
| GUI  | Graphical User Interface                          |
| HTML | Hypertext Markup Language                         |
| IMM  | Intercept Mover Manager                           |
| IEEE | Institute of Electrical and Electronics Engineers |
| JPEG | Joint Photographic Experts Group                  |
| LSVE | Large-scale Virtual Environment                   |
| M&S  | Modeling and Simulation                           |
| MAS  | Multi-Agent Systems                               |
| MOE  | Measure of Effectiveness                          |
| MOP  | Measure of Performance                            |
| NCDS | NGA Commercial Data Service                       |
| NGA  | National Geospatial-Intelligence Agency           |
| NOLH | Nearly Orthogonal Latin Hypercube                 |
| NPS  | Naval Postgraduate School                         |
| PCL  | Property Change Listener                          |
| PNG  | Portable Network Graphics                         |

|        |  |
|--------|--|
| SAVAGE | Scenario Authoring and Visualization for Advanced Graphical Environments |
| SMAL   | Savage Modeling Analysis Language  |
| VRML   | Virtual Reality Markup Language  |
| SRTM   | Shuttle Radar Topography Mission   |
| USGS   | United States Geological Survey  |
| W3C    | World Wide Web Consortium  |
| X3D    | Extensible 3D Graphics Standard  |
| XML    | Extensible Markup Language   |

## **EXECUTIVE SUMMARY**

Almost all flight simulators are centered on the problems that can occur during flight, whereas airport ground traffic problems are seldom addressed and are growing considerably. A number of precautions have been directed by the U.S. Federal Aviation Administration (FAA) to overcome these challenges, such as pilot training and adding taxiway indicator signals to help pilots follow specific paths when taxiing. Further work is needed.

This thesis simulates the problem of Ground Traffic incursions. Discrete Event Simulation (DES) and the Viskit tool are used to build two scenarios describing the takeoff and the landing maneuvers including potential ground incidents. It also presents the different techniques used to build 3D graphics models for the airplanes and the airport environment using Extensible 3D (X3D) graphics. After running the simulation a number of times with different parameters, collected data support basic analysis and potential conclusions. This approach demonstrates a proof-of-concept capability worthy of future work.

THIS PAGE INTENTIONALLY LEFT BLANK

## ACKNOWLEDGMENTS

The author would like to acknowledge the support provided by the Naval Postgraduate School and the Modeling Virtual Environments and Simulation (MOVES) institute. Additionally, the author would like to thank the following people whose direct support made this thesis possible:

My wife Hanen and son Ahmed for their patience, understanding, and support during this project.

Don Brutzman and Arnold Buss for their guidance and support during this thesis research.

Terry Norbraten and Byounghyun Yoo for their willingness to listen and offer the help and the advice on numerous aspects of this thesis research.

John Hiles, Chris and Rudy Darken, Don McGregor, Curt Blais in teaching and for providing me with the required knowledge to perform this research.

My classmates Sevdalin Stoykov and Brian Boone who helped me and supported me during the whole period of study.

THIS PAGE INTENTIONALLY LEFT BLANK

# I. INTRODUCTION

## A. OVERVIEW

This thesis supports the ongoing work of building 3D scenarios in the Modeling Virtual Environments and Simulation (MOVES) Institute of the Naval Postgraduate School. The visualization of the operational scenario in three dimensional (3D) environments is a time consuming-task that requires a specific expertise and broad knowledge on different tools to model and construct 3D scenes. The Savage Library, by offering the use of its growing database of 3D models, helps students elaborate, construct and generate specific scenarios and discover the most advanced techniques in creating these scenarios.

Supporting techniques and technologies include:

- Extensible Markup Language (XML)
- XML Schema
- Extensible Style-sheet Language Transformation (XSLT)
- Extensible 3D Graphics (X3D)
- The Scenario Authoring and Visualization for Advanced Graphical Environment (Savage) Archive of Models
- The Savage Modeling and Analysis Language (SMAL) for tactical Simulation Methodology

This thesis provides a proof-of-concept demonstration of ground-traffic control in order to simulate, and analyze problems from the growing number of accidents occurring during ground maneuver. Future work may be able to add additional necessary features to make the simulation more realistic.



## **B. PROBLEM STATEMENT**

The purpose of adding visualization to a simulator program is to achieve the benefits of observing, analyzing and improving its characteristics to attain a specific behavior. A number of 2D scenes have been added to simulators in order to enrich them, but these 2D scenes are often lacking a number of characteristics that allow the simulation to be more realistic, which is indispensable for many simulations. The example of simulating the taxiing of an aircraft on the runway is one of many simulations that require the realism of 3D visualization. The example of simulating the aircraft on the runway is driven from the purpose of diminishing the number of accidents in Air Traffic control, especially those occurring on the ground. Airlines have agreed to use their cockpit simulators to train pilots to taxi. But most of the existing products of simulators are mainly targeted toward air traffic control not the ground traffic. The demand of constructing simulators for ground traffic control is increasing especially with the growing number of accidents occurring on the ground. The FAA recently decided to require 73 airports to add painted lines on taxiways in order to make it clearer to pilots where they would have to stop to avoid intruding on runways. By itself this measure is not very promising to guarantee the elimination of airplane ground incursions <http://www.faa.gov/runwaysafety/stats.cfm> (Accessed March 2008)

## **C. MOTIVATION**

All airline pilots take regular training exercises in simulators, but most of these simulators treat events that occur during flight and airport approach, such like engine failure, mechanical problems, and severe weather conditions. Now, after a spate of recent cases in which airplanes nearly collided on runways, simulator training focused specifically on taxiing has become an urgent task for pilots in order to prevent such incursion accidents. The FAA website has stated that a number of meetings for about 40 representatives of airlines, airports, and air traffic controllers have been held in order to list the specifications on a new simulators that would help train the pilots and air traffic controllers to better manage the ground traffic of airplanes before takeoffs and after

landings. Information about runway and taxiway safety, as well as precautions taken by FAA to decrease these accidents is available at:

<http://www.faa.gov/runwaysafety/stats.cfm> (Accessed March 2008).

This field of research is complex and presents a good set of material to be implemented and added to the growing number of scenarios in Savage. These scenarios are using a variety of techniques in order to be simulated and visualized in 3D virtual environments.

#### **D. OBJECTIVES**

This research attempts to build scenarios that generate different possible ground-traffic situations for airplanes that include:

- Airplane movement on the ground.
- Communication of the pilots with the control tower.
- Airplane takeoff procedure.
- Airplane landing procedure.
- Collection of data from different possible scenarios to help clarify the best approach to follow to decrease the risk of ground incursion.
- Manipulating 3D editors mainly Blender to build different models of airplanes.
- Learning various techniques to build the airport, runways and control tower.
- Adding SMAL META DATA to models.
- Adding models into Savage Archive.
- Building the different scenarios using Viskit tool.
- Collecting data in order to present a statistical overview of different possible situations.

## **E. THESIS ORGANIZATION**

Chapter II summarizes the related work from previous thesis that applied the same approach in simulating and generating visualizations in three dimensional environments. Chapter III presents the approach taken to build the simulation with Discrete Event Simulation (DES) and X3D Graphics, which are the main techniques used in this approach. It also covers design principles used in constructing the models, as well as a presentation of the different tools used to build the final scenario: Simkit, Viskit, Diskit, and SavageStudio. Chapter IV describes the approach of implementing the model using Simkit/Viskit as well as the approach taken to build the different event-graphs and their translation to the assembly editor. Chapter V describes the procedure to generate the terrain of the airport in X3D format as well as the creation of the airplane model, its exportation to X3D format and their implementation in Savage Archive. Finally, Chapter VI is dedicated to experiment the performance of simulation and to collect data. These data are analyzed and commented through the analysis report generated from the Viskit tool. Conclusions and recommendations for future work are the subjects of the last chapter.

## **II. BACKGROUND AND RELATED WORK**

### **A. INTRODUCTION**

This chapter provides a conceptual description of the technologies and related work that helped and inspired the completion of this thesis. The following sections are intended to provide the reader with the basic resources, explaining how they are used in this thesis. Most information given in these sections is referenced appropriately for further investigation into each subject area.

### **B. DISCRETE EVENT SIMULATION (DES)**

The study of any dynamic system requires a good representative model that mimics the behavior of the actual system. A good model is characterized by its accuracy, realism (how close is the model to the actual system), and the capability to adapt to changes.

Discrete Event Simulation (DES) is a methodology to simulate dynamic system based on a chronological sequence of events. Each event occurs at an instant in time and marks a change of state in the system. The DES process is based on events, state variables, and scheduling edges. The simulation starts with the first event in the event list, and then other scheduled events are added as the simulation progresses. The time advance of the simulation is characterized by the scheduled events in the event list. An event schedules another event with specific conditions and time delay. States change value at particular instants of time named events, which explains the term “Discrete Event.”

#### **1. Event Graph**

An event graph is a graphical representation of a DES model based on events and scheduling edges. An Event Graph consists of nodes and directed edges. Each node represents an event, and each edge represents a scheduling relationship between events.

Each edge can have an associated boolean condition and/or a time delay. Figure 1 depicts a fundamental event graph representation and the event graph and is interpreted as follows:

When Event A takes place, Event B is to be scheduled after a time delay of  $t$ , providing a condition (i) is true (after the state transitions for Event A have been performed). The time delay  $t$  is indicated toward the tail of the scheduling edge and the edge condition is shown just above the middle of the scheduling edge. The basic Event Graph paradigm contains only two elements: the event node and the scheduling edge with two options on the edges (time delay and edge condition).

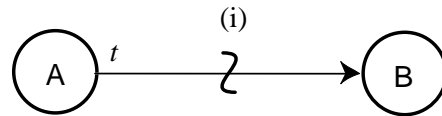


Figure 1. Fundamental event graph representation

## 2. Arrival Process Example

One basic example of Event Graph is the Arrival Process, which is a model with two events (Run and Arrival) and a single state variable, which is the number of arrivals ( $N$ ). The time between arrivals or interarrival times is  $\{T_A\}$ , that can be constant or a sequence of independent and identically distributed random variables. The state transition for the Arrival event is that the number of arrivals ( $N$ ) is incremented by 1 each time the arrival event occurs. The Event Graph for the Arrival Process is shown in Figure 2.

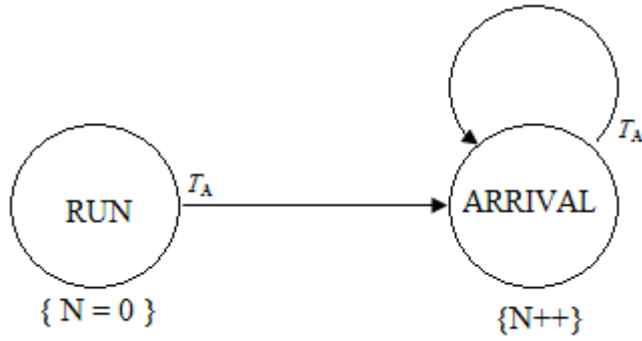


Figure 2. Simple event graph representation (Buss 2001)

Figure 2 represents a simple event graph composed of circles and arrows, where circles are used to represent events and the arrows represent the scheduling edge. The other components of the figure are labels to represent the transition and the scheduling edges (Buss 2001).

### C. ROLE OF 3D VISUALIZATION IN DISCRETE EVENT SIMULATION

The importance of the visualization of a DES model into 2D or 3D environment is crucial for understanding many simulations. Many articles have been written in this matter and more specifically about what the 3D visualization can add to a DES model, since typically DES models are visualized and represented mainly in 2D environments. A survey was conducted about this feature (Akpan & Brooks 2005) and the results of this survey show how important the visualization of a DES model can be. Many useful insights are gained from 3D visualization, and numerical analysis. The 3D visualization of a DES model can inform the researcher about the environment and may lead to a further understanding of the model itself. This paper concluded that, when feasible, modelers are more likely to apply 3D visualization for their DES models.

### D. X3D GRAPHICS

More than 30 years of innovative research, theoretical mathematics, and application development effort have made 3D graphics an exciting field that produces amazing results and frequent surprises (Brutzman and Daly, 2007).

**X3D** is the ISO standard XML-based file format for representing 3D computer graphics and presents the successor to the Virtual Reality Modeling Language (VRML). X3D features extensions to VRML (e.g. Humanoid Animation, NURBS, GeoVRML etc.). Encoding the scene using an XML syntax enhanced application programmer interfaces (APIs).

X3D defines several profiles (sets of extensions) for various purposes, such as X3D Core, X3D Interchange, X3D CAD, X3D Geospatial and X3D Immersive. Browser makers can define their own extensions prior to submitting them for standardisation by the Web3D Consortium. Information about X3D can be found at <http://www.web3d.org> (Accessed March 2008).

## 1. X3D-Edit Authoring Tool

The X3D-Edit 3.2 Authoring Tool for Extensible 3D (X3D) Graphics supports the creation, checking, display and publication of X3D scenes. It is written in open-source Java and XML using the Netbeans platform, which gave the features of being suitable as standalone applications and as a plug-in module for the Netbeans integrated development environment (IDE). X3D Graphics is the tool used for the elaboration the creation of the different model in his thesis. Additional information can be found at <https://savage.nps.edu/X3D-Edit> (Accessed March 2008).

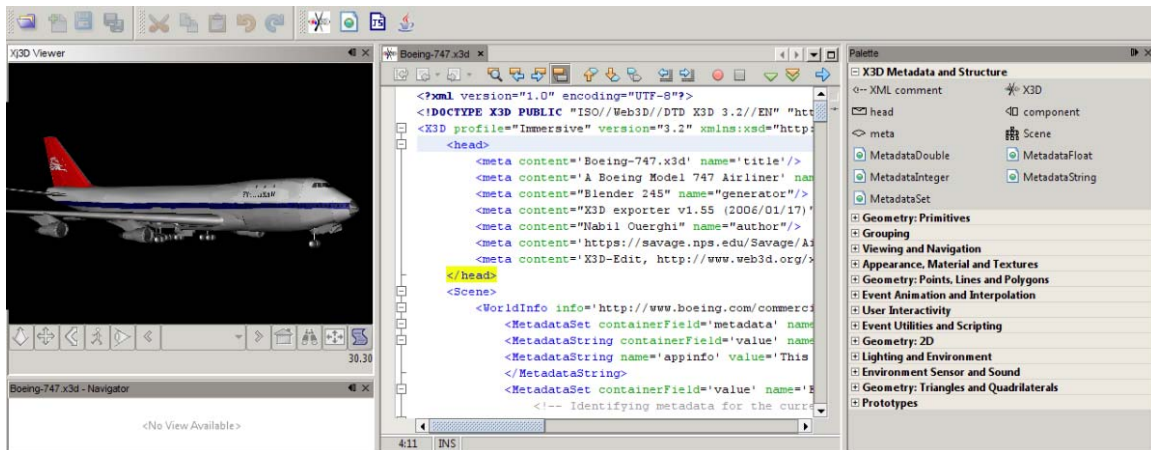


Figure 3. A snapshot of the X3D-Edit user interface for authoring X3D models showing a model of airplane via XJ3D browser.

## 2. XJ3D Open Source Project For X3D

Xj3D is an open source project of the Web3D Consortium Source Working Group focused on creating a toolkit for VRML97 and X3D content. It is written completely in Java. It serves a dual purpose of being an experimental code base for trying out new areas of the X3D specification and as a library. Additional information on Xj3D can be found at <http://www.xj3d.org> (Accessed Feb 2008).

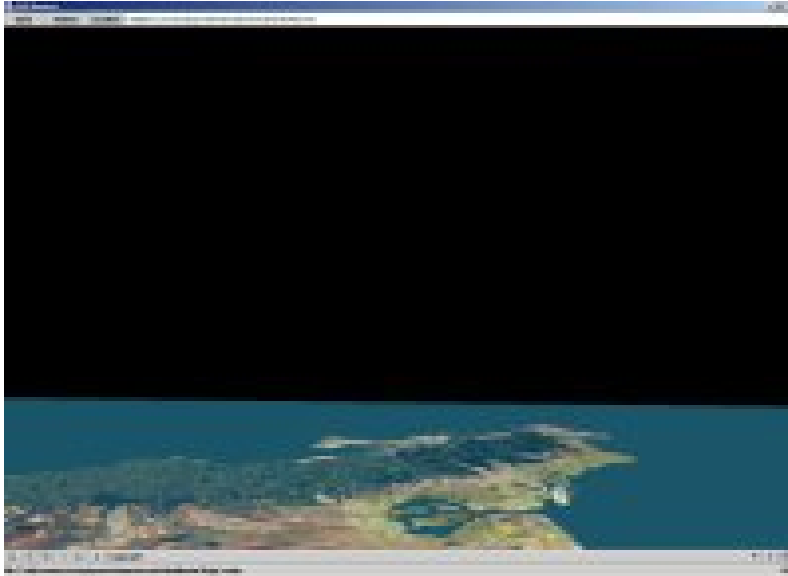


Figure 4. A snapshot of the Xj3D browser showing a model of terrain in X3D format.

## 3. VIZX3D / FLUX Studio Scene Authoring Tool

FLUX STUDIO (former Vizx3D) is graphic tool for X3D that offers the capability to create, manipulate and visualize 3D models. It has many features that allow the user to import their model from different graphic tools such as Blender or Wings3D. It also has many features indispensable for animation and scripting, as well as its own 3D browser for displaying 3D object visualization. Additional information on FLUX STUDIO can be found at <http://www.mediamachines.com> (Accessed February 2008).



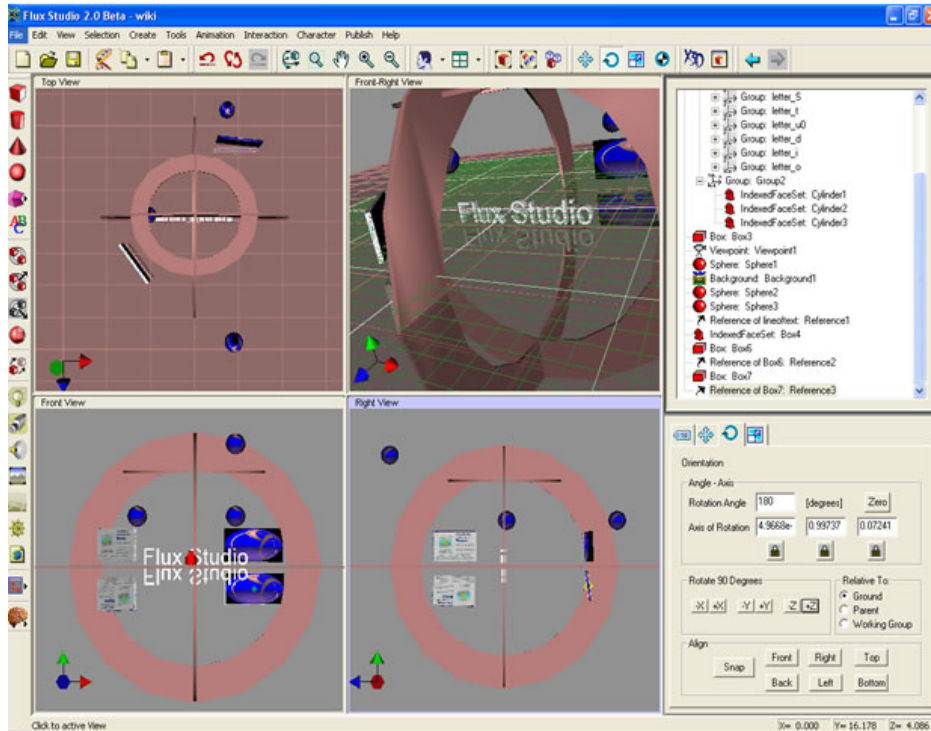


Figure 5. Flux Studio user interface for authoring X3D models.

#### 4. WINGS3D Authoring Tool

Wings3D is an open source modeling program which highlights fast workflow for creating polygons, meshes and complex geometries of virtual objects. It allows users to model, manipulate, and export 3D models in a variety of formats as for example, it can export into VRML format so the model can be exploited and modeled in other 3D editing tools such as X3D-Edit . Its interface is easy to understand and fast to start with as well as the possibility to work with without having a deep knowledge in the domain of 3D modeling. Wings 3D is usable by beginners in 3D modeling. Additional information on Wings 3D can be found at <http://www.wings3d.com> (Accessed February 2008)

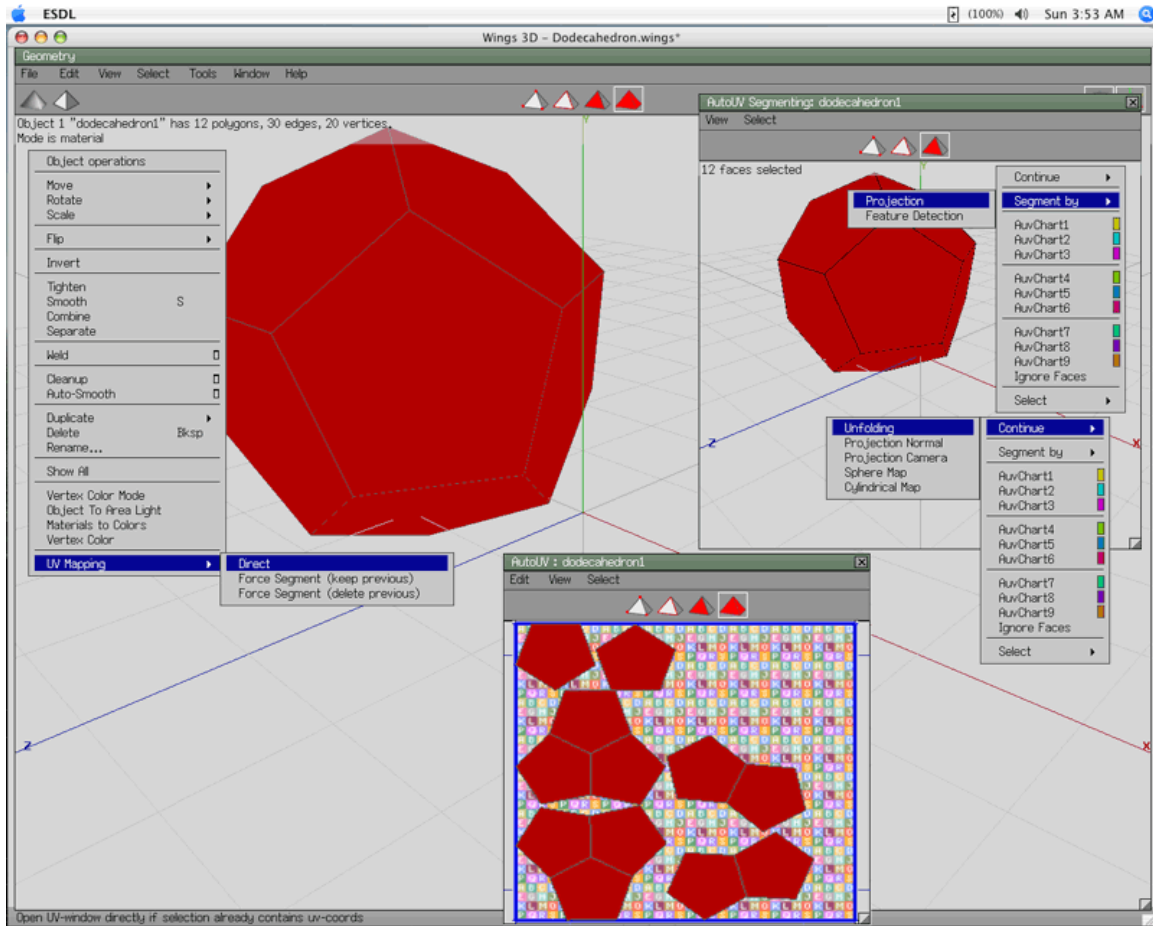


Figure 6. Wings 3D User Interface for creating 3D models that can be exported to X3D format.

## 5. Blender Authoring Tool

Blender is a free open source graphics editor mainly used for creating 3D models. It offers a variety of tools to generate highly defined models. Blender is also used for generating animation that can be added to the models, as well as scripts to make the models interactive. Its interface is complicated but there are video and text tutorials that may help the student to effectively use this tool.

Blender is one of the 3D modeling tools recommended for research studies, because it offers a number of features as for example:

1. Open source
2. Character animation support (CAL3D)
3. Game engine with physics and visual programming

Additional information on Blender can be found at <http://www.blender.org> (Accessed February 2008).

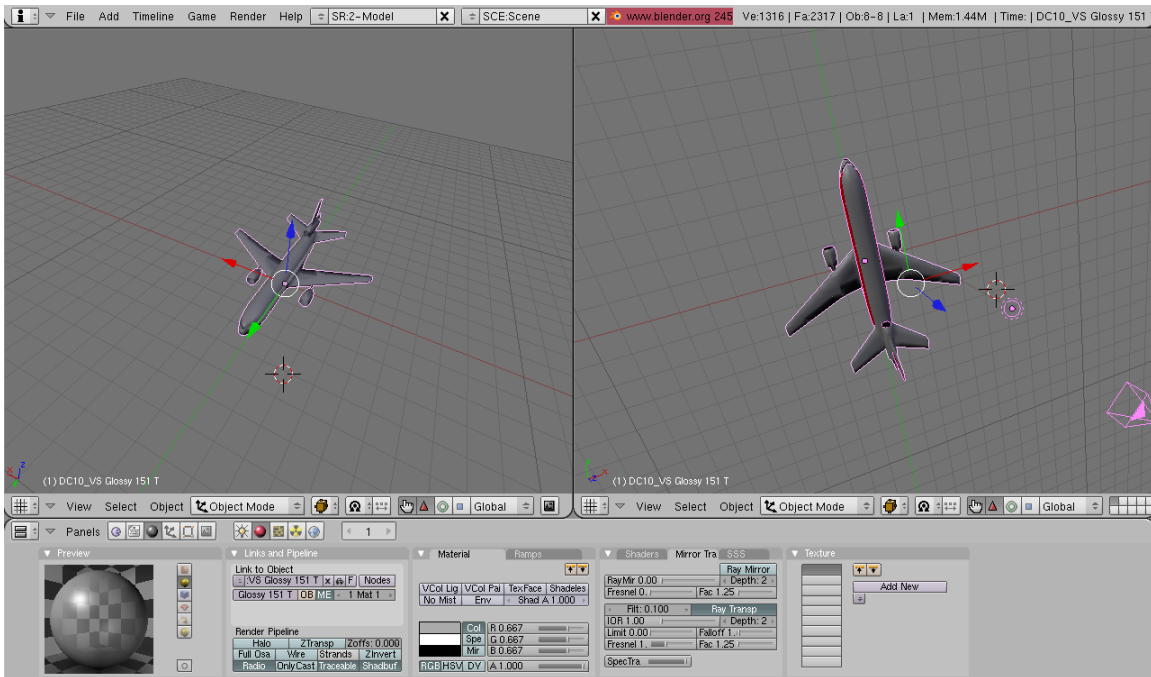


Figure 7. The Blender interface showing different views of an airplane construction.

## E. IEEE DISTRIBUTED INTERACTIVE SIMULATION (DIS) NETWORK PROTOCOL

Distributed Interactive Simulation (DIS) is an open standard that helps interconnect multiple host computers to realize a real-time platform-level of communication like for example, in War gaming. This technology is used in worldwide in many domains especially for military purposes as well as in medicine.

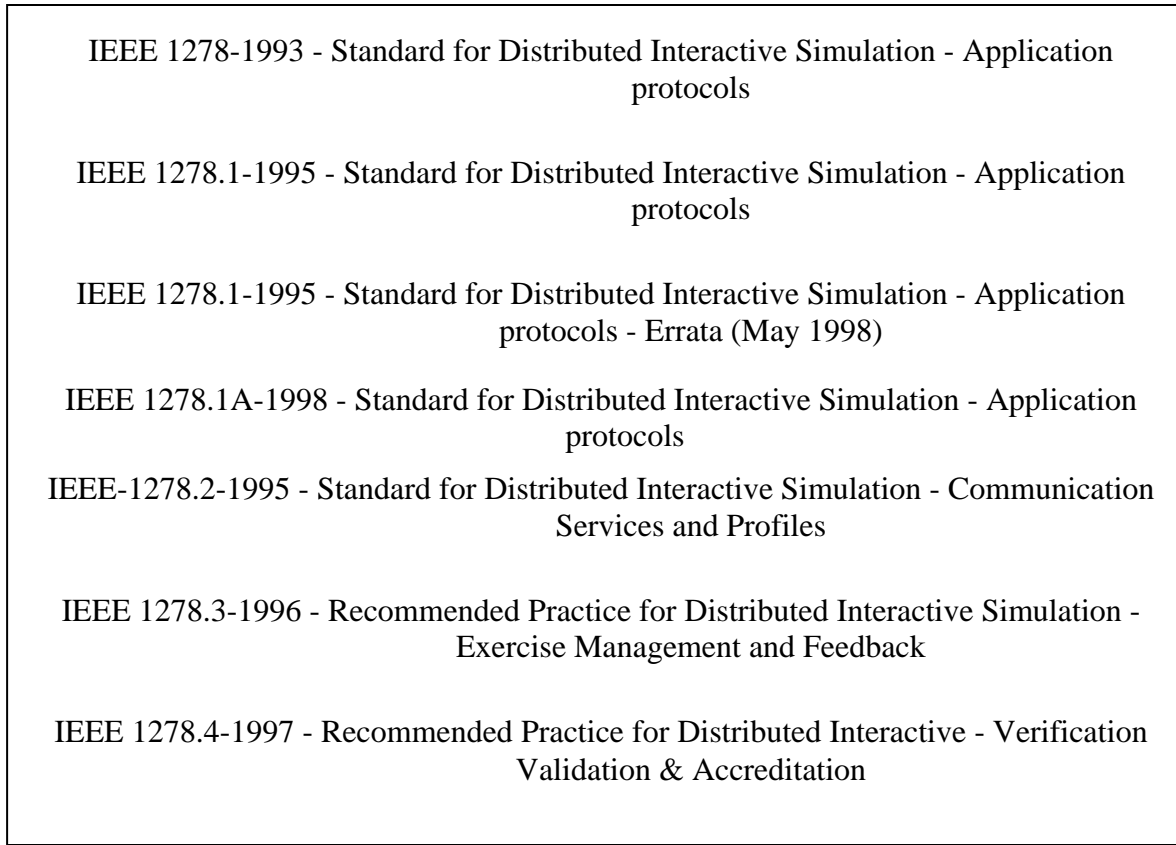


Figure 8. The Distributed Interactive Simulation (DIS) family of standards.

Additional information about DIS is available at the site <http://ieeexplore.ieee.org> (Accessed March 2008).

**F. SAVAGE MODELING ANALYSIS LANGUAGE (SMAL) FOR SIMULATION METADATA**

Savage Modeling and Analysis Language (SMAL) is the Metadata strategy for identifying tactical, physical and simulation-oriented Metadata for vehicles, terrain and entities in virtual environments. The main component of a simulation other than the coding part is the visualization task, which is difficult to achieve when the visualization is in 3 dimensions. SMAL offers the possibility to more quickly assemble the environment for such simulation, and handle all the components of the simulation. It also exposes the NPS 3D model archives for easy integration and use in almost all simulation purposes.

Additional information about Savage (Scenario Authoring and Visualization for

Advanced Graphical Environments) are in the site <https://savage.nps.edu/Savage/Tools/SMAL/SMAL.html> (Accessed Mars 2008).

## **G. SUMMARY**

This chapter has presented the different technologies that are related to the accomplishment of this thesis, as well as some previous research in the same area. A number of 3D graphic editors is presented in this chapter Readers are encouraged to follow the links in every subject for further acknowledgement.

### **III. BEHAVIOR MODELING CAPABILITIES**

#### **A. INTRODUCTION**

The purpose of the thesis is to produce a visualization of DES model of an Airport ground traffic control. Different tools are used in order to create the different components, which are the models, the code and the environment. Graphics tools are used to create the airplane's model, imagery and airport elevation data are made to generate the terrain, and Simkit/Viskit is used to create the code of the simulation. This chapter covers the steps followed in building the scenarios of the simulation in Viskit tool.

#### **B. DISCRETE EVENT PROGRAMMING WITH SIMKIT**

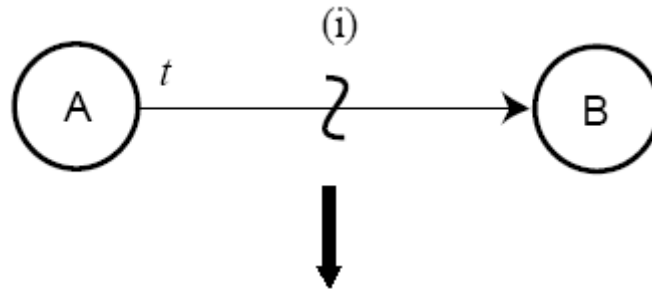
Simkit is a software package for creating Discrete Event Simulation (DES) models written in Java. It adapts a DES approach that provides flexibility and modeling power then. Simkit implements Future Event List (FEL) in a class called `simkit.Schedule` consisting of static methods and variables; the FEL is responsible for the time advance of the simulation and is hidden from the simulation modeler (Buss 2001).

In this thesis, Simkit was used to translate the model of airplane ground traffic control to DES approach and more specifically into event graph representation.

##### **1. SimEntity and SimEntityBase**

The `SimEntity` is an interface specified by a set of methods that must be implemented by any class that is using the FEL and other simulation objects.

The `SimEntityBase` is an abstract class that implements most of the functionality for interacting with the FEL. The basic example of the event graph modeling two events A and B presented in Figure 1, is translated in to java class that extends the abstract class `SimEntityBase` and decode the events as shown in Figure 11.



```

public void doA() {
    <code to perform state transition for event A>
    if (i) {
        waitDelay("B", t);
    }
}

```

Figure 11. The code associated to the state transition for event A in the basic event graph example.

The code shown in Figure 11 translates the state transition for the event A, which schedules event B under the condition (i) and a time delay t. The events are characterized by methods that have “do” in front of the event name such as doA. The waitDelay() method takes two arguments, which are the name of the event scheduled and the time delay associated with scheduled event, it creates an instance of SimEntity that has a reference to object that created it (Buss 2001).

## 2. Simple Movement and Detection in DES

The scenario being considered in this thesis deals with movement, which is traditionally modeled using time stepped manner. Since the model is created in DES approach with Simkit software, the problem of the time advance of the simulation is solved. In a DES model, the entity in motion is characterized by an implicit state of position determined from three explicit states, which are:

- The entity’s position when it started to move
- The time it started to move
- Its velocity vector

The simplest possible movement is uniform, linear motion. A moving entity starts its move at some initial position  $x$  at time  $t_0$  and begins moving with velocity  $v$ . Thus, the location of the entity at time  $t$  is  $x + (t - t_0) v$ . This is a simple movement of the entity from point A (origin) to point B (destination). A more complex rules of movement is one with a list of waypoints and collisions detections, the approach implemented in this case is the use of detection and undetection techniques

The solution to the equation (1) given when the sensor detects an enter range movement gives two possible cases which are EnterRange event and the ExitRange event, the EnterRange event schedules the detection procedure with a delay given by a process such as for example an exponential delay .

$$t = -\frac{x.v}{\|v\|^2} \pm \frac{\sqrt{\|v\|^2(R^2 - \|x\|^2) + (x.v)^2}}{\|v\|^2} \quad (1)$$

Equation (1) gives the two points of enter and exit range of the sensor.

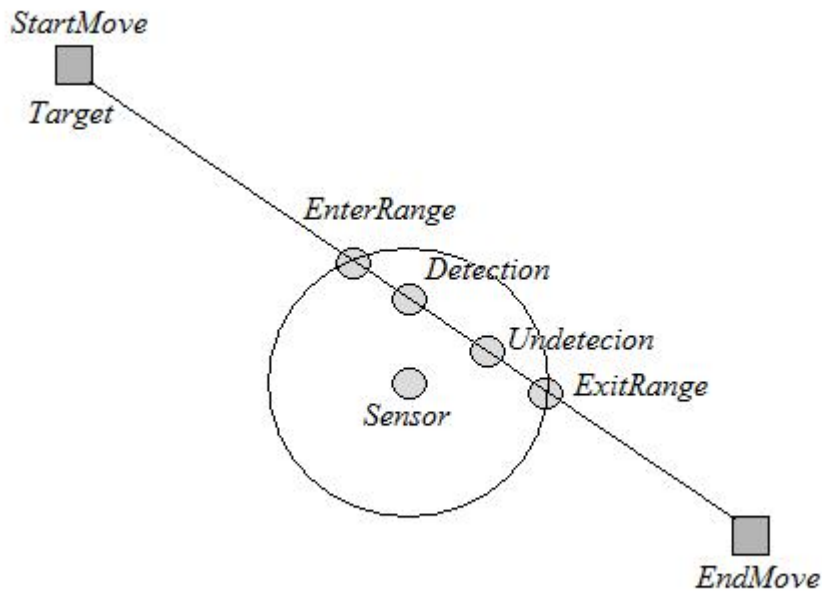


Figure 12. The moving entity detected by the sensor just after it enters the sensor range and undetected just before exiting the sensor range(Buss and Sanchez 2005).



### 3. Mover3D and DISMover3D

Mover3D is a java interface that specifies the minimum requirement of any implementing classes that deal with interactions such detection and undetection.

DISMover3D implements the Mover3D interface and extends SimEntityBase, it provides all the required functionalities of a 3D mover to detect other object and send its state as DIS packets across the network (Seguin 2007). Figure 13 depicts the DISMover3D event graph in Viskit tool, which is the discussed in the next section.

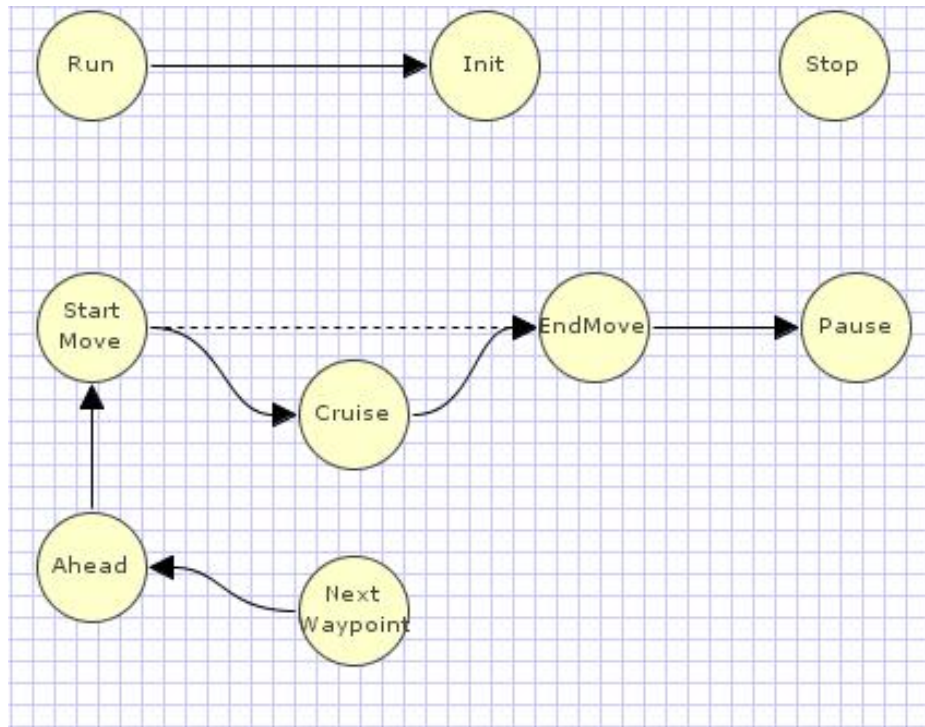


Figure 13. Illustration of the event graph showing the different components of the DisMover3D (Sullivan 2006).

### C. EVENT GRAPH EDITOR IN VISKIT

Viskit is an application for defining event graph models; these models can be used for different purposes, such as multiple server queues simulation and the Anti-Terrorism Force Protection (ATFP) simulation. This tool is used to model the airplane ground traffic simulation.

The application allows building event graph models that describe the behavior of the entities that define the components in the simulation. Figure 14 shows the user interface for Viskit tool.

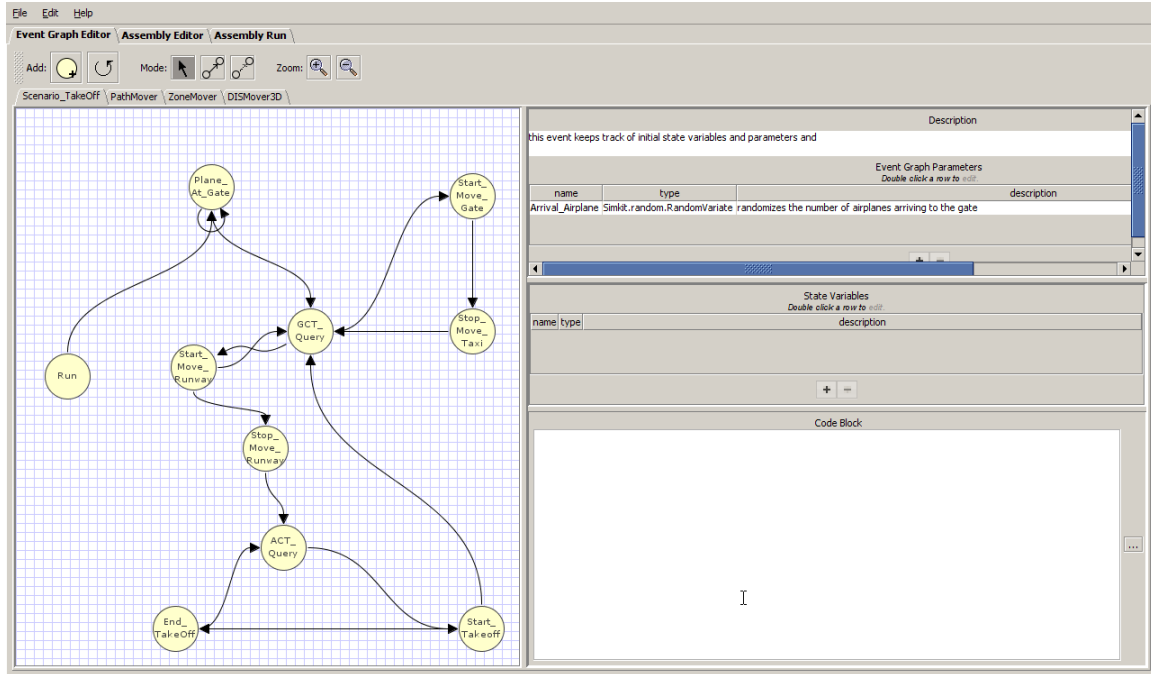


Figure 14. A snapshot of Viskit tool showing the user interface with the different tools.

## 1. Event Graph Parameters

The event graph shown in Figure 14 is composed of circles (events) and arrows (scheduling edges). It offers an easy approach by inserting events and connectors between these events. This feature of Viskit tool makes the concept of building the components of the simulation easier and less time-consuming. The only precaution to consider is the parameterization of the events with the proper state variables and arguments. The scheduling edges as an element of the event graph, has also a specific configuration which consists of the Boolean conditions and/or a time delay. And to better present the different components of the Viskit tool, a basic example of the arrival process will guide the readers through building and configuring the event graph in Viskit. A representation of the event graph depicting the Arrival process is shown in Figure 2. To

create a new event graph in Viskit, first, Viskit tool must be launched via Netbeans or from the executable file downloadable from <http://diana.nps.edu/Simkit> (Accessed March 2008). This site illustrates how to add Simkit in the library of Netbeans or Eclipse. After starting Viskit tool, go to *File> New event graph*

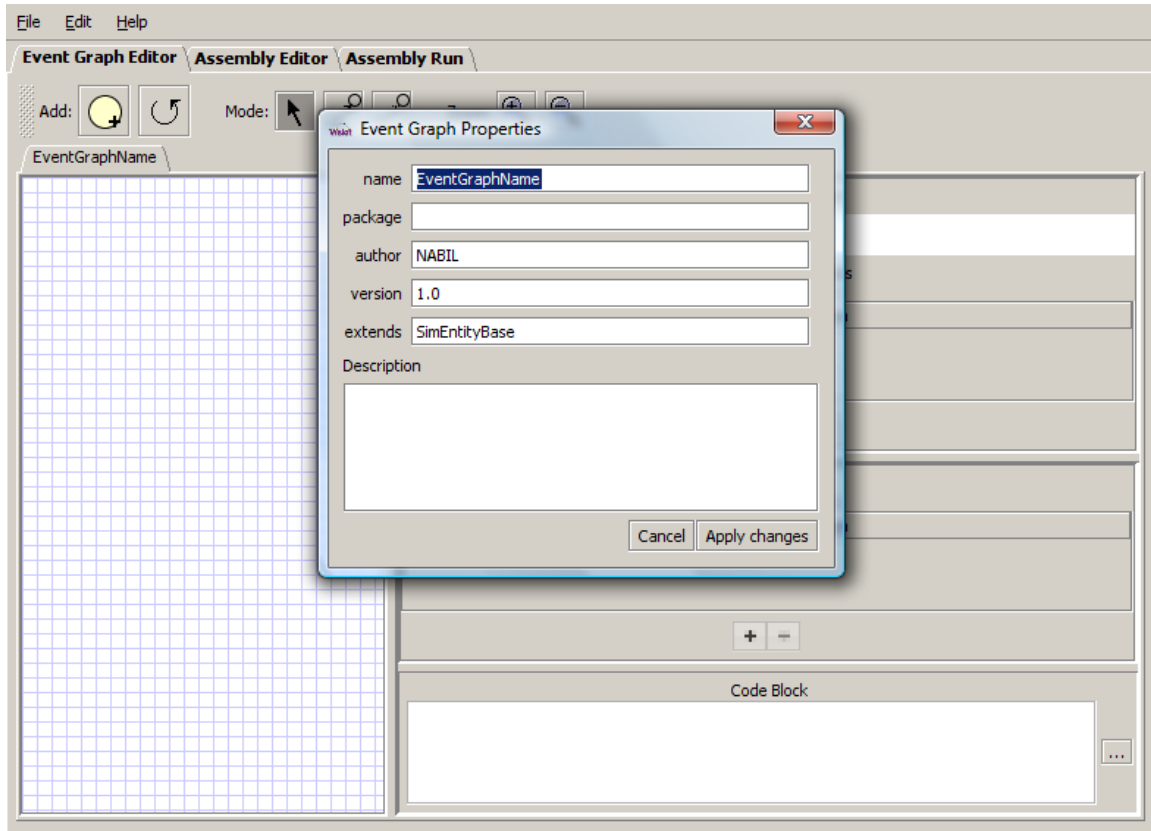
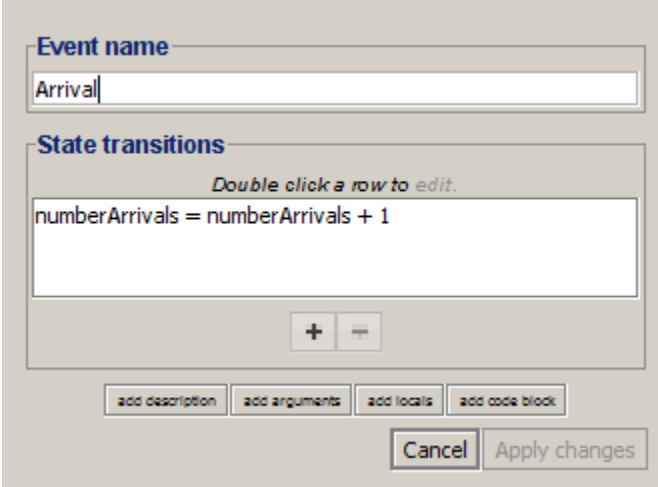


Figure 15. Snapshot depicting the procedure of creating a new event graph with Viskit application.

As is it shown in Figure 15, by creating a new event graph a menu pops up to enter the characteristics of the event graph such as the name, the package, the author, and the version. The author can drag the circles (events) and the arrows (scheduling edges) to create the event graph.

## 2. Defining Event Parameters

By double clicking on an event, a menu pops up to enter the parameters of the event. It is also possible to parameterize the scheduling edge in the same way by double clicking on it.



The screenshot shows a dialog box for defining event parameters. It has a title bar and a main content area. At the top, there is a section titled "Event name" with a text input field containing the word "Arrival". Below this is a section titled "State transitions" with a text area containing the code "numberArrivals = numberArrivals + 1". Above the text area is the instruction "Double click a row to edit." Below the text area are two buttons: a plus sign "+" and a minus sign "-". At the bottom of the dialog are four buttons: "add description", "add arguments", "add locals", and "add code block". At the very bottom are two buttons: "Cancel" and "Apply changes".

Figure 16. The menu in which the author can parameterize the event.

## D. ASSEMBLY EDITOR IN VISKIT

### 1. Creating a Simple Assembly

After the event graph is parameterized, tested through the compilation of the generated code and saved, the author switches to the assembly editor in the Viskit tool. A new menu opens where the event graph models already created are stored into the user specified directories. Figure 17 shows the menu of the assembly editor where in the up left there are the directories with the saved event graph models.

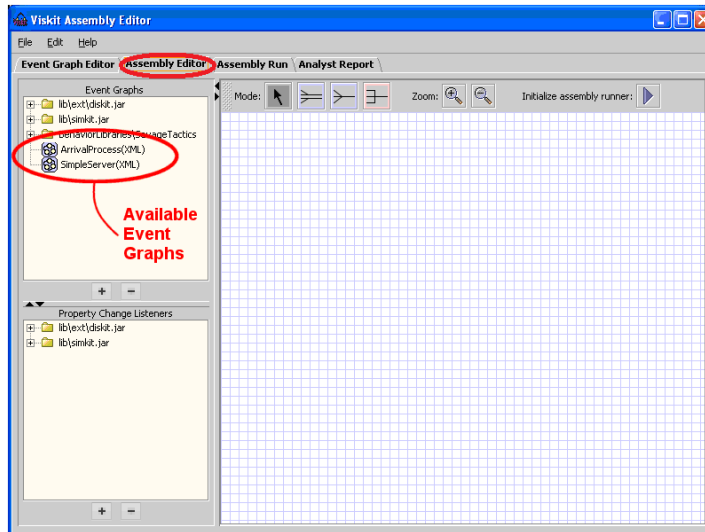


Figure 17. Assembly editor in the Viskit tool where the scenario author can use an already-created event graph.

Similarly to the event graph, the author can configure the new created assembly in the menu associated with it by filling out the parameters such as the name, the package, the version, and the author as shown in Figure 18.

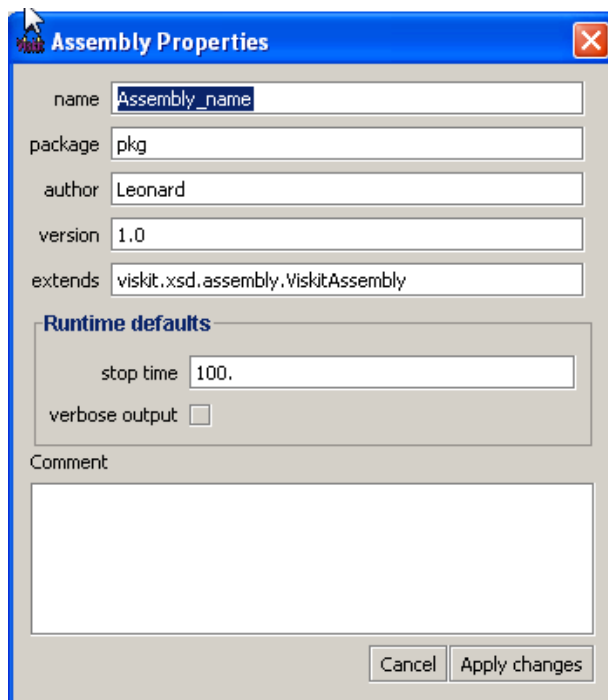


Figure 18. The menu allowing the author to configure the new created assembly.

Figure 19 shows how the author can manipulate the created event graphs in the assembly editor by a simple operation of drag and drop.

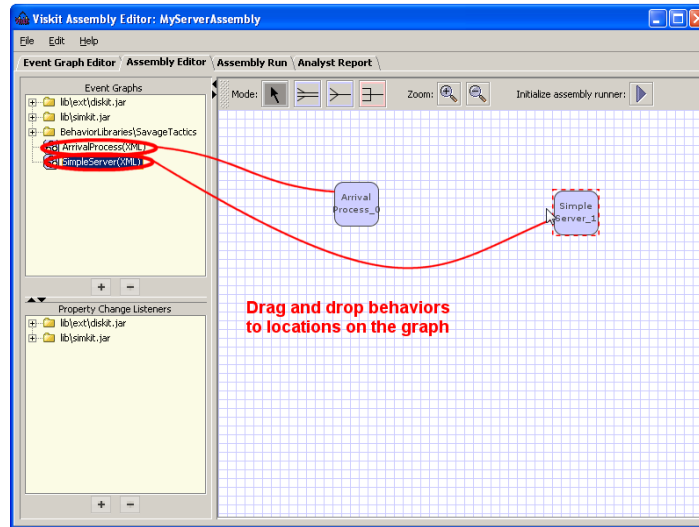


Figure 19. The Manipulation of the saved event graph models in the assembly editor menu.

In order to connect the different events, the author can chose from the different type of connector that are explained in Figure 20

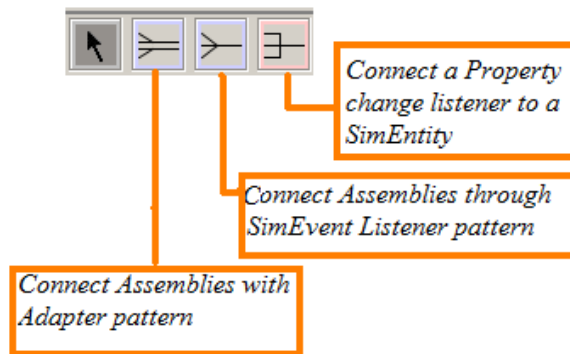


Figure 20. The different connectors in the Assembly Editor with their different properties.

## 2. Property Change Listener (PCL)

As its name indicates, the function of Property Change Listener (PCL) in Viskit, when hooked to an event graph model, is to listen to the changes occurring on the state variables and records these changes. Depending on the type of the PCL, the collected data is output in different format to the author. Viskit offers various types of Property Change Listeners that are shown in Figure 21.

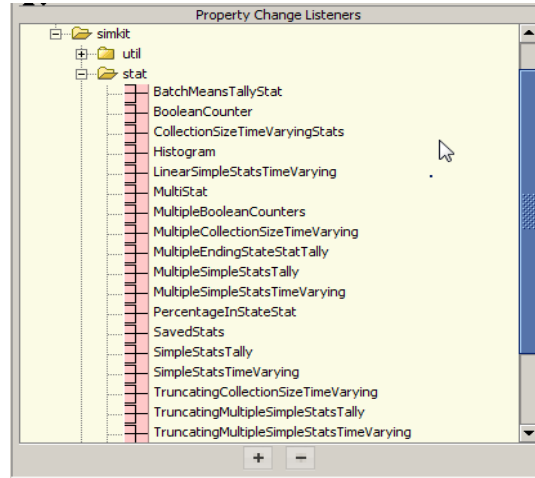


Figure 21. Different Property Change Listeners (PCL) in the menu of Assembly Editor.

| Property Change Listener          | Parent package | Description  |
|-----------------------------------|----------------|--|
| Collection size time varying stat | Stat           | Track the changes occurring in the specified queues              |
| Simple stat tally                 | Stat           | Track changes occurring in the specified state variables         |
| Simple stat time varying          | Stat           | Track the changes in the time of in the specified state variable |
| Simple property dumper            | Util           | Track the changes occurring on the queues                        |

Table 1. The different connectors to the assemblies used for both scenarios

### 3. The Assembly Run

After setting all the parameters for the event graph models in the assembly editor and connecting them with the proper Property Change Listeners, the author saves the changes and switches to the Assembly run menu, where he can configure the run parameters. By clicking on run simulation in the upper right icon in the assembly interface, a new menu appears where the parameters required to run the simulation must be entered. These parameters are:

- The start time of the simulation
- The stop time of the simulation
- The number of replications
- The verbose feature
- A summary report of the simulation

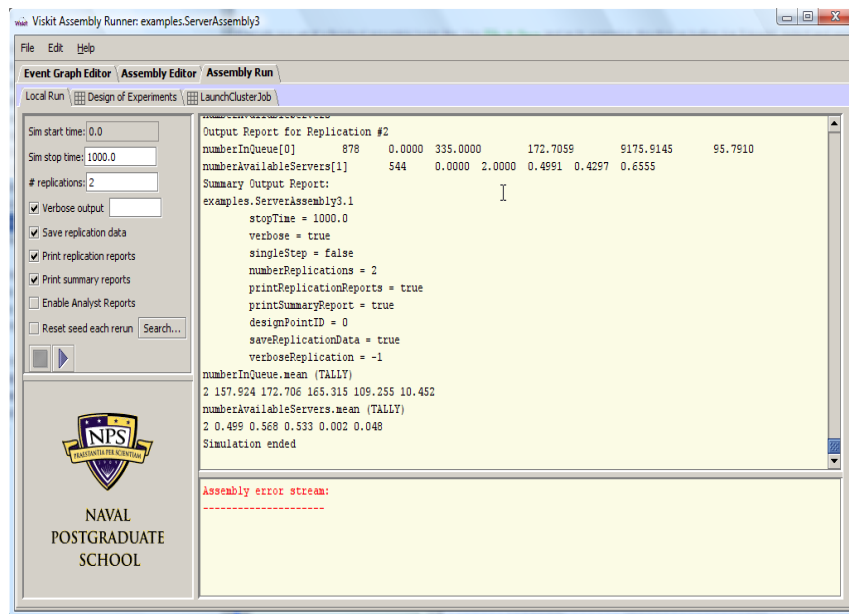


Figure 22. The interface where the author can configure and run the simulation.



## 4 The Assembly Report

Once the parameters of the simulation are configured in the assembly run menu, the simulation is ready to be launched. One important feature of Viskit tool is the assembly report, which is configurable through the menu of the assembly run. After the simulation run and stops, an Analysis report is saved as html page, so that the author can analyze it afterward. Figure 23 depicts a summary of the main tables in the Analysis report generated when the simulation stops running.

**Summary of Simulation Entities**

| Simulation Entity       | Behavior Definitions                    |
|-------------------------|---|
| <a href="#">arrival</a> | <a href="#">examples.ArrivalProcess</a> |
| <a href="#">server</a>  | <a href="#">examples.SimpleServer</a>   |

| Initialization Parameter | Parameter Type              | Description         |
|--------------------------|-----------------------------|---------------------|
| interarrivalTime         | simkit.random.RandomVariate | no comment provided |

| State Variable | Variable Type | Description         |
|----------------|---------------|---------------------|
| numberArrivals | int           | no comment provided |

| Initialization Parameter | Parameter Type              | Description         |
|--------------------------|-----------------------------|---------------------|
| serviceTime              | simkit.random.RandomVariate | no comment provided |
| totalNumberServers       | int                         | no comment provided |

| State Variable         | Variable Type | Description         |
|------------------------|---------------|---------------------|
| numberInQueue          | int           | no comment provided |
| numberAvailableServers | int           | no comment provided |

| Run# | Count   | Min   | Max     | Mean    | StdDev  | Variance  |
|------|---------|-------|---------|---------|---------|-----------|
| 1    | 881.000 | 0.000 | 333.000 | 157.924 | 102.980 | 10604.924 |
| 2    | 878.000 | 0.000 | 335.000 | 172.706 | 95.791  | 9175.915  |

| Entity | Property            | Count | Min     | Max     | Mean    | StdDev | Variance |
|--------|---------------------|-------|---------|---------|---------|--------|----------|
|        | numInQueueStat      | 2.000 | 157.924 | 172.706 | 165.315 | 10.452 | 109.255  |
|        | numAvailServersStat | 2.000 | 0.499   | 0.568   | 0.533   | 0.048  | 0.002    |

Figure 23. Summary of Tables generated by the Analysis Report after the simulation stops running.

### E. SUMMARY

This chapter presents the DES approach in creating the simulation model, as well as a presentation of Simkit package with its different components and how DES is expressed in Simkit. This chapter also covered the main features of the Viskit tool, which

offer tremendous help in programming complex simulations. The Viskit tool generates the program, compiles it, runs it, and analyzes it. In this chapter, the author utilized some basic examples of the event graph models such as the Arrival Process to help illustrate the main features of the Viskit tool. The different steps in building the simulation in Viskit are:

- Building the event graph model.
- Building the assembly associated to the event graph model.
- Configuring and running the simulation.
- Enabling the analysis report and displaying it after the simulation stops running.

THIS PAGE INTENTIONALLY LEFT BLANK

## **IV. MODEL IMPLEMENTATION USING VISKIT**

### **A. INTRODUCTION**

This chapter is dedicated to the presentation of the framework, the approach taken to simulate the different scenarios as well as the tools used to implement these scenarios. The simulation is divided into two scenarios: the takeoff scenario and the final scenario. This approach simplifies the procedure of building the event graph model for each scenario. The scenarios are tested in the assembly run, with different configurations to get the more realistic behavior of the airplanes.

### **B. METHODOLOGY**

To implement the different scenarios, bottom-up approach is followed that starts with the first scenario to be as simple as possible to keep a running state of the simulation. Then, different features will be added to the first scenario as the simulation sophistication progresses.

The first scenario simulates the behavior of the airplane in the Takeoff procedure, while in the second scenario; the behavior of a landing airplane is added to form the final scenario.

### **C. USE CASES**

The first scenario is the simulation of the takeoff procedure, which consists of five stages:

1. Departing airplane at Departure Gate
2. Departing airplane ready to start taxiing
3. Departing airplane at taxiway
4. Departing airplane at runway
5. Departing airplane at flight zone

The second scenario is describing the landing procedure of the airplane and it adds five more stages:

1. Arriving airplane in the Approach Flight Zone
2. Arriving airplane on the Runway
3. Arriving airplane on the Taxiway
4. Arriving airplane at Arrival Gate

#### **D. SCENARIO NARRATIVES**

The scenario narrative is a presentation of the different stages of the different scenarios. In order to move from one stage to the other, a number of possible cases were studied and implemented. For Example, if the query to the Ground Control Tower was sent in the same time by two Different events, such as, Arriving airplane at the Approach Zone and Departing airplane on the taxiway. The first event is given permission to schedule the Start Land event with higher priority, which is close to realistic situation. The airplane landing has higher priority then the airplane preparing for the takeoff. In Figure 24, a detailed presentation of the takeoff scenario is presented. The blue arrows describe the taxiway maneuver, starting from the Gate to the entrance of the runway, which are presented by both blue and yellow arrows. The two Runways are presented in black bold lines, and the red arrows depict the movement of the airplanes from the Gate to the Taxiway entrance.

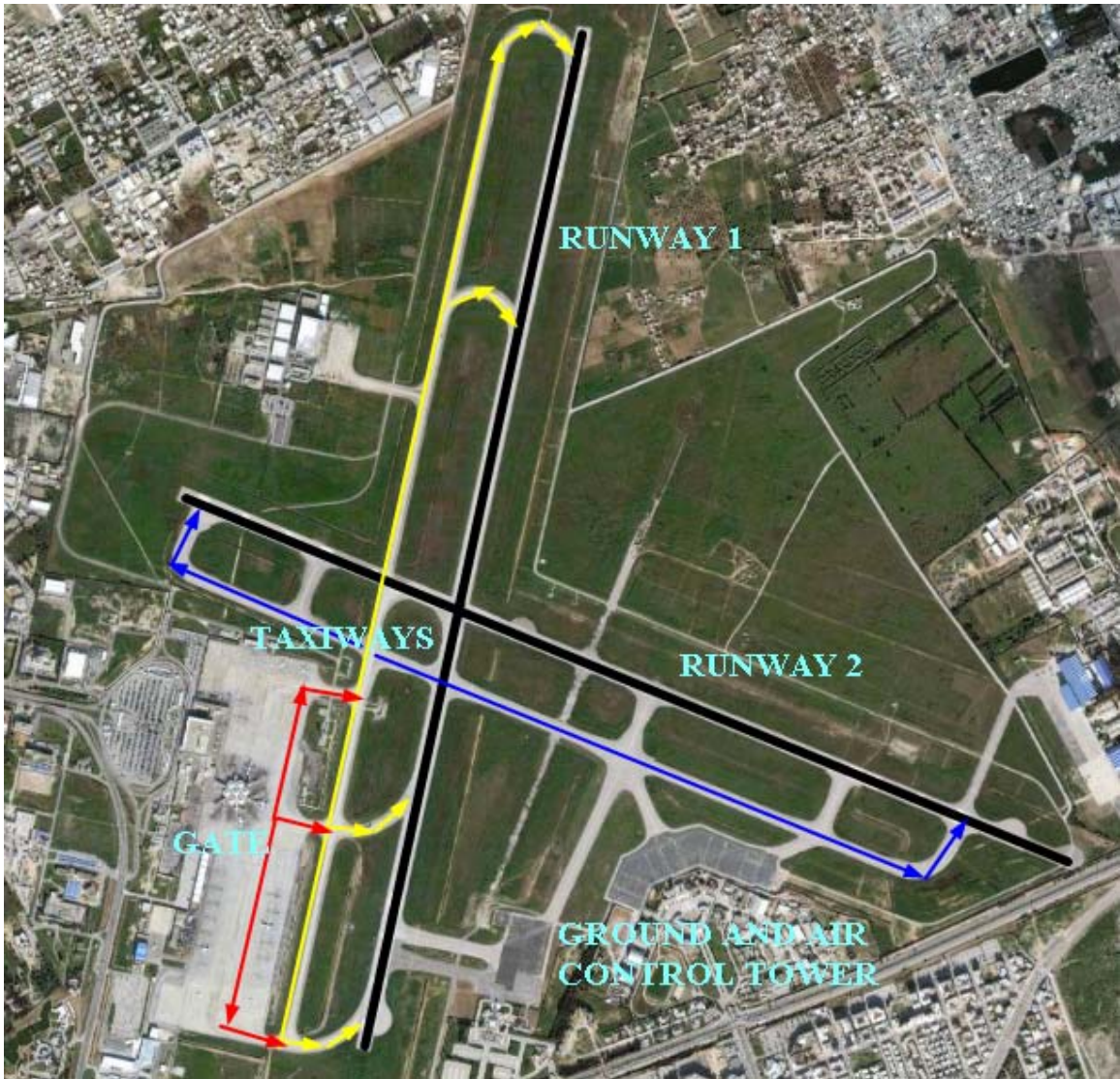


Figure 24. The different stationary points in the Environment of the simulation for the takeoff scenario.

**E. RIGHT OF WAY RULES**

The right of way rules are the rules that specify the priorities between different possible cases that can happen during the takeoff and the landing procedure. The different possible cases where the Ground Control Tower answers the query to simultaneous events are given in Table 2.

| Events                          | Priority |
|---------------------------------|----------|
| Takeoff                         | Highest  |
| Landing                         | Default  |
| Arrival Runway to Taxiway       | Highest  |
| Departure Taxiway to Runway     | Default  |
| Departure Gate to Taxiway       | Default  |
| Arrival Taxiway to Arrival Gate | Highest  |

Table 2. The different possible simultaneous events with their priorities

Table 2 shows the right of way rules between the airplanes while changing their states. The procedure of takeoff starts from the departure gate to the Departure flight zone going through taxiway and runway. The landing procedure starts from the Arriving approach zone to the Arrival gate going through the runway and the taxiway.

## F. ENTITIES AND DATA STRUCTURES

The different components of the model were presented by different data structure types. These data structure were objects of type:

- Simkit.Entity: identifies and indexes the airplanes in the different queues.
- Linked List: the linked list was used to express the FIFO (First In First Out) procedure of storing the airplanes before moving to the next events.
- Simkit.random.RandomVariate: The time delay that the airplane spends to move from one state (event) to another is expressed in exponential distribution with different mean values.

Table 3 summarizes the different data structure types used to model the different components of the simulation.

| Entity                    | Data Structure type  | Description  |
|---------------------------|--|--|
| Airplane                  | simkit.Entity  | Objects of type Simkit.Entity to differentiate between different airplanes |
| timeTogatewayDeparture    | simkit.random.RandomVariate<br>An Exponential distribution with specified mean | InterArrival time for the airplane to come to the gate                     |
| timeToReadyDeparture      | simkit.random.RandomVariate<br>An Exponential distribution with specified mean | Time for the airplane to get ready to move in the Departure Gate           |
| timeToTaxiwayDeparture    | simkit.random.RandomVariate<br>An Exponential distribution with specified mean | Time for the airplane to go taxiway  |
| timeToRunwayDeparture     | simkit.random.RandomVariate<br>An Exponential distribution with specified mean | Time for the airplane to get to the Runway                                 |
| timeToFlightZoneDeparture | simkit.random.RandomVariate<br>An Exponential distribution with specified mean | Time for the airplane to get to the flight zone                            |
| timeToApproachZoneArrival | simkit.random.RandomVariate<br>An Exponential distribution with specified mean | Time for airplane to come to the Approach zone                             |
| timeToRunwayArrival       | simkit.random.RandomVariate<br>An Exponential distribution with specified mean | Time that takes the airplane to land                                       |
| timeToTaxiwayArrivals     | simkit.random.RandomVariate<br>An Exponential distribution with specified mean | Time that takes the airplane to go from Runway to Taxiway                  |
| timeToGateArrivals        | simkit.random.RandomVariate<br>An Exponential distribution with specified mean | Time that takes the airplane to go to the Arrival Gate                     |
| queueDepartureGate        | Java.util.LinkedList   | Contains the airplanes entities at the Departure gate                      |



|                           |                      |  |
|---------------------------|----------------------|--|
| queueReadyDeparture       | Java.util.LinkedList | Container for the airplanes entities getting ready to move |
| queueTaxiawyDeparture     | Java.util.LinkedList | Container for the airplanes entities at the Taxiway        |
| queueRunwayDeparture      | Java.util.LinkedList | Container for the airplanes entities at the Runway         |
| queueFlighZoneDeparture   | Java.util.LinkedList | Container for the airplanes entities at the Flight zone    |
| queueArrivalGate          | Java.util.LinkedList | Container for the airplanes entities at the Arrival gate   |
| queueTaxiwayArrivals      | Java.util.LinkedList | Container for the airplanes entities at the Taxiway        |
| queueRunwayArrivals       | Java.util.LinkedList | Container for the airplanes entities at the Runway         |
| queueApproachZoneArrivals | Java.util.LinkedList | Contains the airplanes entities at the Approach Zone       |

Table 3. Different simulation components with their following data structure type

## G. EVENT GRAPH MODELS FOR THE TWO SCENARIOS

### 1. First Scenario

The entire simulation was divided into two sub-scenarios, this approach made the final scenario easier and flexible for any eventual changes:

- Takeoff scenario
- Takeoff scenario + Landing scenario = Final scenario

Once the two scenarios were implemented in Viskit, it was easy to connect them together to form the final scenario. The first scenario, which is the take off scenario, is

composed of 12 events connected together with scheduling edges. The logic followed in configuring the scheduling edges is explained in Table 4. Specific Time delays and passing arguments were added to the scheduling edges and events in the event graph model.

| Events                           | Scheduling events                                  | Description  |
|----------------------------------|--|--|
| Run                              | Arrival at Departure Gate                          | Initialize all state variables to zero.  |
| Arrival at Departure Gate        | Departure airplane ready                           | The airplane is ready after check and passengers load procedure is finished.   |
| Departure airplane ready         | Query the GCT to Taxiway                           | Asking permission from the Ground Control Tower to Taxiway.  |
| Query the GCT to Taxiway         | Go to Taxiway                                      | After checking the statue of the taxiway the permission is given.  |
| Go to Taxiway                    | Departure airplane At Taxiway                      | This procedure will define the time for the airplane to move to the taxiway.   |
| Departure airplane At Taxiway    | Query the GCT to Runway                            | Asking permission from the Ground Control Tower to Runway.   |
| Query the GCT to Runway          | Go Runway  | The permission is given to move to the Runway under the condition of the runway.   |
| Go Runway                        | Departure airplane at the Runway<br><br>Go Taxiway | This event schedule two events airplane at Runway when permission is given and Go Taxi to let other airplanes coming from the Departure Gate to enter the Taxiway. |
| Departure airplane at the Runway | Query the ACT to Takeoff                           | For the first scenario the permission is given under no condition.   |

|                                       |                                       |   |
|---------------------------------------|---------------------------------------|---|
| Query the ACT to Takeoff              | Go Takeoff                            | For the first scenario the permission is given under no condition.  |
| Go Takeoff                            | Departure airplane at the flight zone | This event is to finish the taking off procedure with a time delay specified by the user.   |
| Departure airplane at the flight zone | Query the GCT to Runway               | After the airplane takes off it schedules a query to GCT to check if there are airplanes on the taxiway waiting to access the runway. |

Table 4. The different events composing the First Scenario

Figure 25 summarize the different events forming the first scenario, which are explained in Table 4.

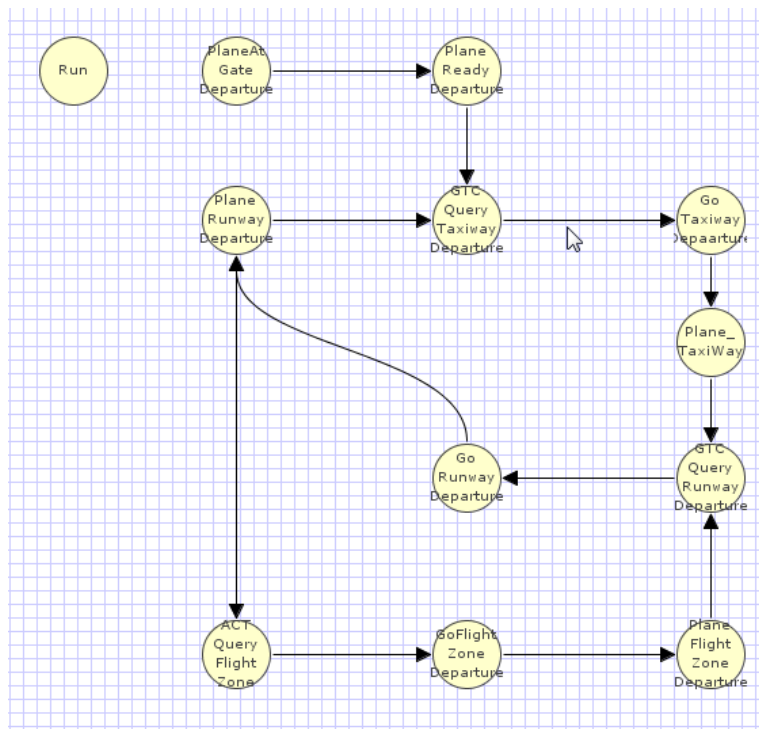


Figure 25. The event graph of the Takeoff scenario showing the different stages the airplanes go through before taking off.

## 2. Final Scenario

The second scenario is the final scenario for the arriving and departing airplanes, and consists of the first scenario with added features, events and scheduling edges. The main feature added to the first scenario is the ability to control the number of airplanes arriving and departing. This feature allows the control of the traffic quality depending on the season and the flight frequencies. For example, in Tunisia, the summer season presents the highest traffic rate for both departures and arrivals flights, which can be configured properly using the added feature to the program. The final version of the program has some other features added to the Takeoff scenario, such as some extra tests for the statue of the runway and taxiways. The number of airplanes departing and arriving is fixed to a maximum of 3. Figure 24 shows the Taxiway capacity as well as the entrance to the taxiways from the different runways. The procedure of landing is described in Table 5.

| Events                      | Scheduling events             | Description  |
|-----------------------------|-------------------------------|--|
| Run                         | Arrival at Approach Zone      | Initialize all state variables and queues to zero                                |
| Arrival at Approach Zone    | ACT query for landing         | The airplane is ready for landing and waiting for permission to land             |
| ACT query for landing       | Start Landing                 | When a permission is given to land a start landing events starts                 |
| Start Landing               | End landing                   | The End landing event will be triggered with a time delay and queues are updated |
| End landing                 | GCT query to the Taxiway      | The airplane after landing will be prompted to access the taxiway                |
| arrival airplane at taxiway | GCT query to the Arrival gate | Ask permission to move to Arrival Gate   |

|              |           |   |
|--------------|-----------|---|
| Arrival Gate | Query GCT | When the airplane arrives at the arrival gate the statue of the queues is updated and another request for the airplane on the taxiway is send |
|--------------|-----------|---|

Table 5. The procedure of landing translated into event graph structure

Table 5 shows the events added to the first scenario describing the landing procedure. The final scenario groups the takeoff scenario and the landing procedure scenario in one event graph. This approach of building the simulation in one event graph makes the tests on the condition of the runways and the taxiways shared by both departing and arriving airplanes, more consistent. For example, the test on the state of the runway for the arriving and departing airplanes must be done on both queues:

$$\text{QueueDepartingRunway.size()} + \text{QueueArrivalRunway.size()} == 0$$

This test is done by both arriving and departing airplanes with higher priority for the arriving airplanes. Figure 26 depicts the final scenario implementation in event graph model.



## H. ASSEMBLY MODELS FOR FIRST AND FINAL SCENARIOS

After building the different events in the event graph, and before running it through the assembly run, it is advisable to test the event graph syntax through the compilation test. The Viskit tool generates the code associated to the event graph and compiles it. In case there are errors, the compiler informs the author about these errors to allow debugging the event graph producing the program. This one important feature of the Viskit tool prevents the author from going to the assembly run with an incorrect event graph model. The next step is to switch to Assembly interface in which the user will find the already built event graph models in the directories where they are saved. With drag and drop operations, the event graph is associated to Property Change Listener (PCL) from the library. This procedure is explained in details in Chapter V.

### 1. First Scenario

In the first scenario, the feature of controlling the number of airplanes taking off is not included; hence there is a need to add other elements to trigger the generation of entities (airplanes) to the “airplanes at departure gate” event. These components are Arrival process and creator.

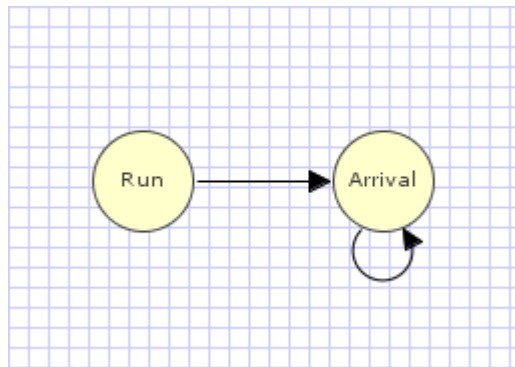


Figure 27. Simple Arrival Process event graph responsible for incrementing the arrival of airplanes.

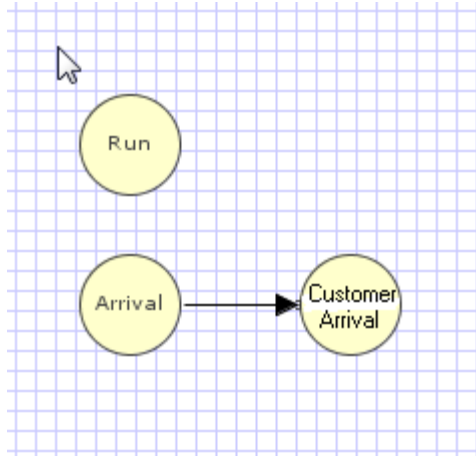


Figure 28. Creator event graph responsible for creating airplane entities.

Figures 27 and Figure 28 depict the implementation of the different event graph models of simple arrival process and the creator. In the Assembly editor the user can drag and drop these three event graphs from the library.

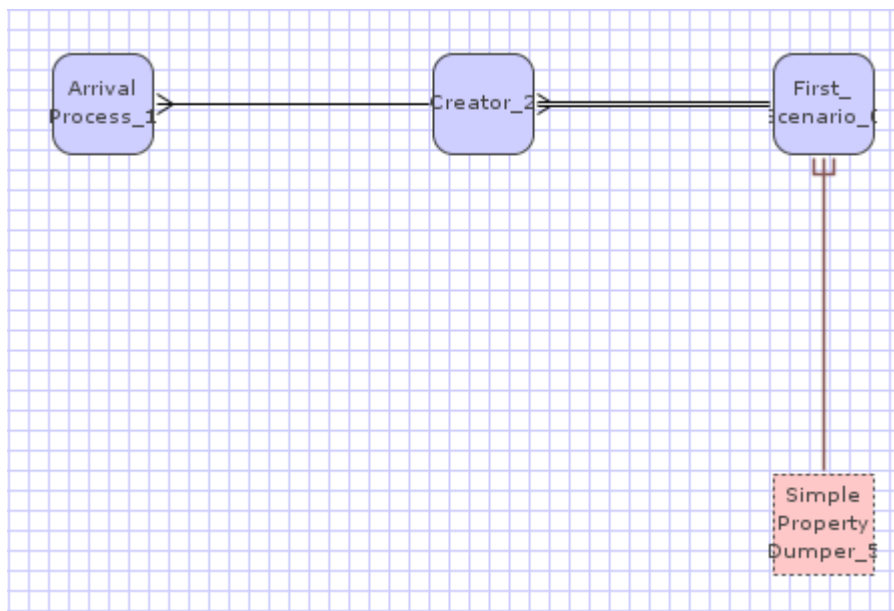


Figure 29. Assembly of the Takeoff scenario with a simple property dumper to track the different linked lists containing the airplanes.

The configuration of the assembly run associated with the takeoff scenario is done by inserting different values of mean to the exponentials distributions associated with the time delays of the scheduling edges. Associating other distributions to represent the time



delays in event scheduling is an adoption for future work. Table 6 summarizes the components of the assembly as well as the connectors used to connect these components.

| Assembly components        |                            | Connector type | Description   |
|----------------------------|----------------------------|----------------|---|
| Arrival Process            | Creator                    | Listener       | The creator is listening to the Arrival Process event graph.  |
| Creator                    | First scenario event graph | Adaptor        | The adapter has two parameters to configure the source event and the target event, in this case, the arrival process is the source and the arrival airplane at Departure gate will be the target. |
| First scenario event graph | Property Change Listener   | Listener       | The user is prompted to specify which state variable he is monitored.   |

Table 6. Description of the assembly of the first scenario.

## 2 Final Scenario

The final scenario has the feature of controlling the number of airplanes taking off and landing. These numbers are predefined before the run of the simulation. This characteristic of the program makes it more realistic, since it might match the daily plan for arriving and departing airplanes. The traffic rate also is not the same for the final period of the year, in some countries the traffic rate depends on the season; the summer presents the highest traffic rate in the year, which is the case in the Tunisia Carthage Airport.

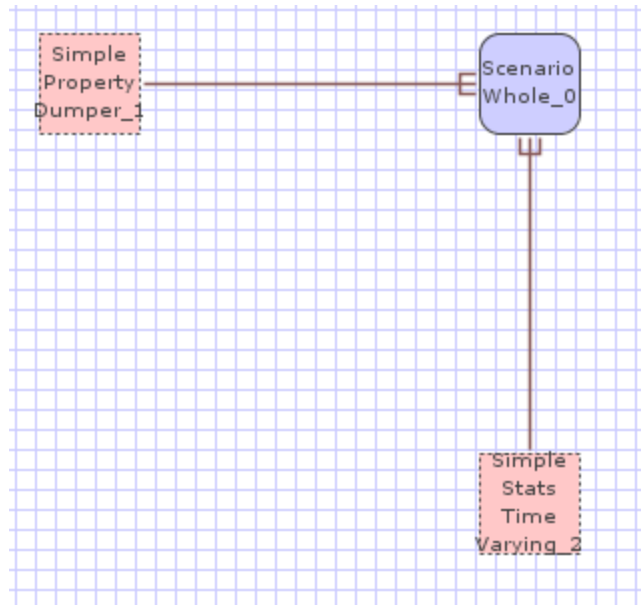


Figure 30. Assembly of the final scenario with two property change listeners.

The difference between the takeoff scenario and the landing scenario is that in the takeoff scenario, there is no parameter specifying the number of airplanes taking off. In the final scenario, this feature is added. The final scenario assembly has fewer components than the first scenario because the complexity is treated inside the event graph model. Table 7 summarizes the assembly associated with the final scenario event graph model.

| Assembly components |                           | Connector | Description   |
|---------------------|---------------------------|-----------|---|
| Final scenario      | Simple property Dumper    | Listener  | The Simple property Dumper listens to the specified state variables                     |
|                     | Simple Stats Time Varying | Listener  | The Simple Stats Time Varying tracks the frequencies of changes occurring on the queues |

Table 7. Different components forming the final scenario assembly.

## **I. MEASURE OF EFFECTIVENESS (MOES) FOR BOTH SCENARIOS**

Measures of effectiveness (MOE) for both scenarios are obtained by running and collecting data from each scenario. The user may specify the nature of the Property Change Listener he wants connect to the Event graph components in the assembly. A summary of the results of each scenario is in the Appendix A and B. The generated code for the different event graph models is also included.

## **J. SUMMARY**

This chapter explained the different procedures of takeoff and landing scenarios and their translation to event graph structure. A bottom-up approach was followed in this simulation. The modeling approach started with the simplest possible scenario describing the takeoff procedure for one airplane, features were then added step by step. These features included augmenting the number of the airplanes and resolving the problem of congestion on the taxiway. After the takeoff scenario was finalized and tested for various parameters, the next step was to build the landing scenario with the same approach as the takeoff scenario. The landing scenario was tested apart with one airplane, then saved compiled and run for different parameters. The last step was to combine the takeoff scenario and the landing scenario together in one event graph model and add extra tests for components of the simulation that are shared by both scenarios, such as the runway and the taxiways. Once the final scenario was build and tested for various parameters, the author conducted analysis by running the simulation for three arriving airplanes and three departing airplanes, using different values of the mean for the Exponential distributions and for 100 replications for the simulation. The results are consistent and the simulation is finalized.

## **V. VISUALIZATION OF THE GROUND TRAFFIC WITH X3D GRAPHICS**

### **A. INTRODUCTION**

The simulation of the ground traffic of the airplanes includes the environment in which the simulation takes place, as well as the model that represents the different entities of the simulation. The environment is Tunisia Carthage airport with the runways, the taxiways and the gate. The other component of the simulation to be modeled is the airplane that represents the entities performing the different tasks of the model. This chapter covers the technique used to collect imagery and elevation data for the creation of the airport in X3D format, as well as the tools used to model the airplane Boeing 747 and export it to X3D format. Finally, the last section of this chapter covers the implementation of the different components in the master scene.

#### **1. Data Collection**

The physical environment in which the simulation occurs is Tunisia Carthage airport. In order to build the airport, specific data must be gathered consisting of imagery and elevation data. A search of public data servers was made, but few of the accessed servers provided sufficient detailed data for the terrain and facility imagery. Figure 31 shows the Global Mapper user interface used to download data from different public servers. <http://www.globalmapper.com> (Accessed March 2008).

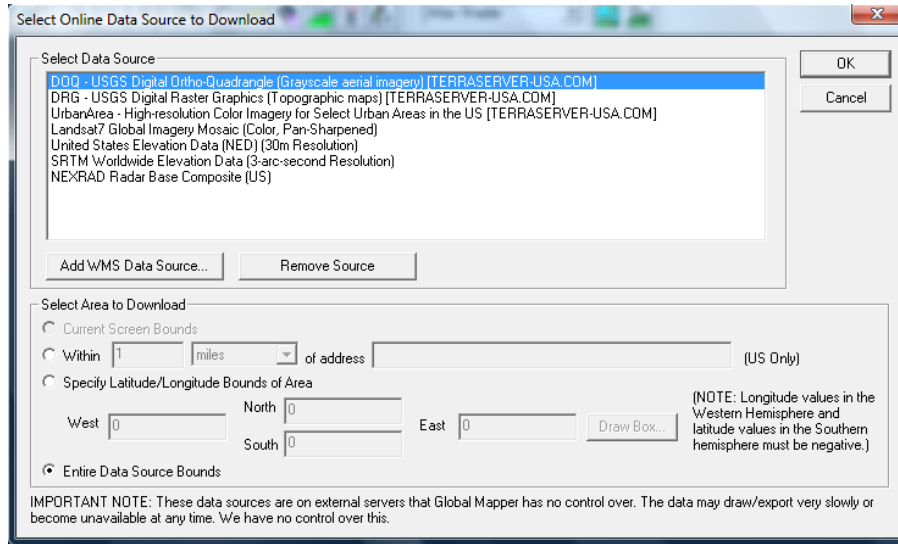


Figure 31. Available online data sources for imagery and elevation using Global Mapper.

**a. Imagery Data**

The available data for imagery were retrieved from the National Geospatial Intelligence Agency (NGA) Commercial Data Service NCDS. The NCDS server provides commercial data for almost every region in the world. The only difficulty was the access requirements specified by the NCDS to be able to download these data. The NCDS offers their services to any military establishment for free. Since NPS qualifies the imagery data were downloaded to NPS from this server. Further information about NCDS is available at:

[https://ncds.nga-earth.org/nga/resources/help/ncds\\_portal.htm](https://ncds.nga-earth.org/nga/resources/help/ncds_portal.htm) (Accessed March 2008).

**b. Elevation Data**

The USGS (United States Geological Survey) provides both U.S. and international data, but the highest quality data are focused on U.S. regions. The data for other regions of the world are available but the quality of the effectiveness and the coherence of these data sets are not very satisfactory. Since the terrain used in the scene

creation is almost completely flat, there is no need to download a highly detailed elevation data and those available in USGS were sufficient to generate the required fidelity for terrain. For further information about the USGS the reader may refer to: <http://seamless.usgs.gov/tutorial.php> (Accessed March, 2008)

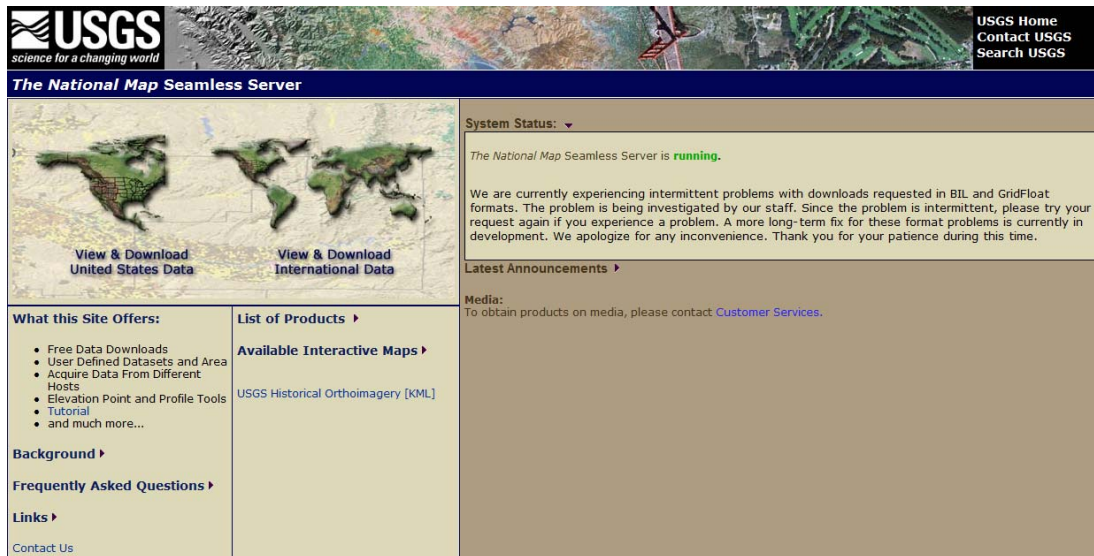


Figure 32. The USGS interface for downloading elevation data.

An end user requesting data of the region of his interest zoom in and adjust the scale of the region, until achieving the coverage of interest. While progressing through the procedure of scaling the region of interest, the author can observe the data available to download in the right bottom of the interface screen. Figure 33 presents the menu the author may refer to observe the availability of the data.

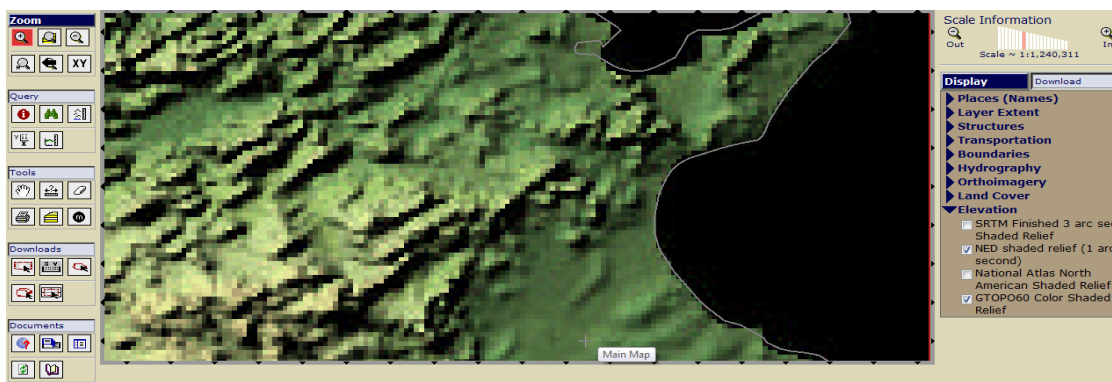


Figure 33. The interface the user may use to specify the region and observe the availability of the data.

The second step after specifying the region and making sure that the data sets of interest are available is to download these data.

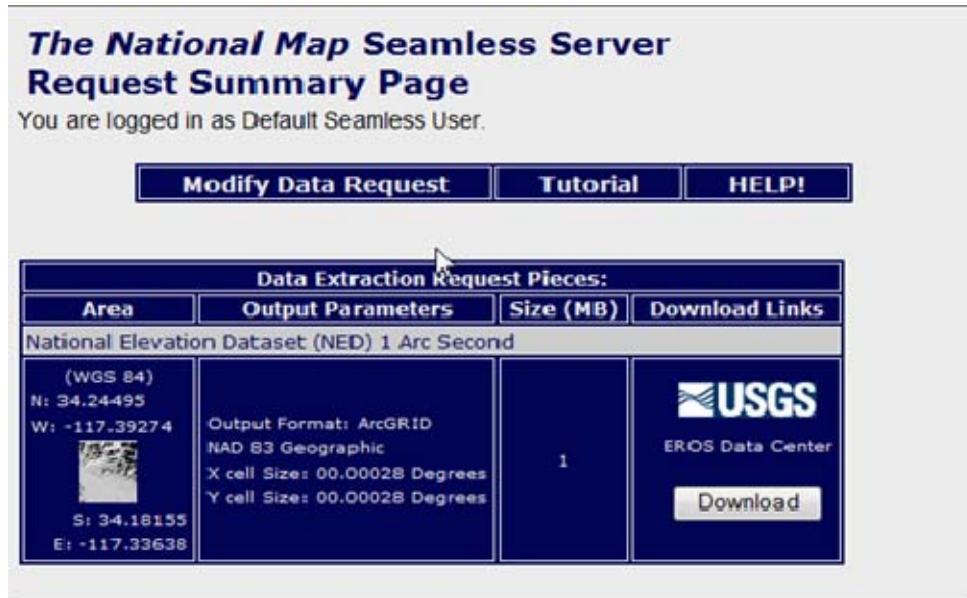


Figure 34. The menu allowing the download the elevation data.

The other way to download elevation data about an international region such as Tunisia is through Global Mapper, especially when there is no available data on USGS server. The Global Mapper software offers the possibility to choose which type of data to download; here the data of interest is the elevation data. Shuttle Radar Topography Mission (SRTM) server provides the necessary data for the region of Tunisia Carthage airport.



Figure 35. The user interface of Global Mapper used to download the airport elevation data.

After specifying the data to download in the menu showed in Figure 35, another menu pops up where the coordinates of the region of interest are entered and the format of the data to download is specified. In the present case the format of the data is ASCII format.

#### *c. Terrain Generation*

After downloading imagery and elevation data, a script to generate tiles for the X3d scene is used. The script is ASCII2GeoX3d and is available at <https://savage.nps.edu/svn/nps/Savage/Locations> (Accessed March 2008). This script launches Rez application to generate the terrain model.

#### *d. Rez Tool*

Rez is an Open source application written in Java for translating gridded data, mainly geospatial into different formats including images and multiresolution models for X3D or VRML web browsing. Rez generates a main scene file, called



Display.wrl/x3d/x3dv, usually with walk/fly tours and useful viewpoints. For more information about Rez tool the reader may refer to <http://www.rez3d.com> (Accessed March 2008). After running Rez tool with the specified inputs, the X3D scene of Tunisia Carthage airport is generated. This scene requires addition SMAL Metadata specific for the terrain model. The specification of SMAL Metadata for the terrain models is defined in the thesis (Rauch 2007) and available at:

<https://savage.nps.edu/Savage/Tools/SMAL/SavageTerrainMetadataTemplate.x3d> (Accessed March 2008).

Figure 36 shows the menu that appears when the Rez application is launched through the script.

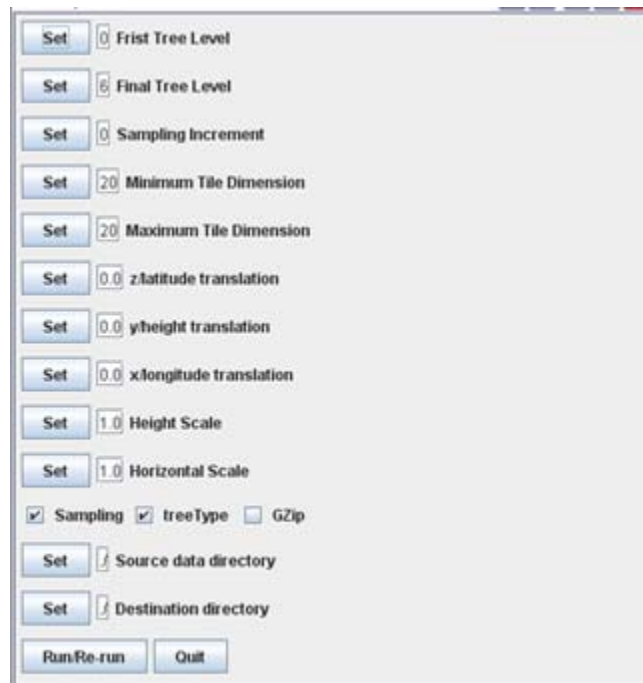


Figure 36. The menu that appears when the Rez application is launched.

Figure 37 shows the 3D model of Tunisia Carthage airport generated with Rez application and displayed in XJ3D browser.



Figure 37. The 3D terrain of Tunisia Carthage airport generated by Rez and displayed with XJ3D browser.

## 2. Building the Airplane

To build the model of the airplane, a number of 3D graphic tools are available and they are summarized in Chapter II. One of the tools that offer the possibility to export any model to X3D format is Blender, which offers many features that make the modeling task easy and less time consuming. It also offers the possibility to apply a background image, which can be the blueprint of the airplane. Specifying different views of object being modeled helps shaping the model and fitting it with the blueprint. Blueprints for different types of objects are available at:

<http://www.the-blueprints.com> (Accessed March 2008)

The airplane modeled is the Boeing 747. Its blueprint is downloaded from the site previously cited. Figure 38 shows the Blender interface with three different views and three blueprints showing the airplane in front, top and side view.

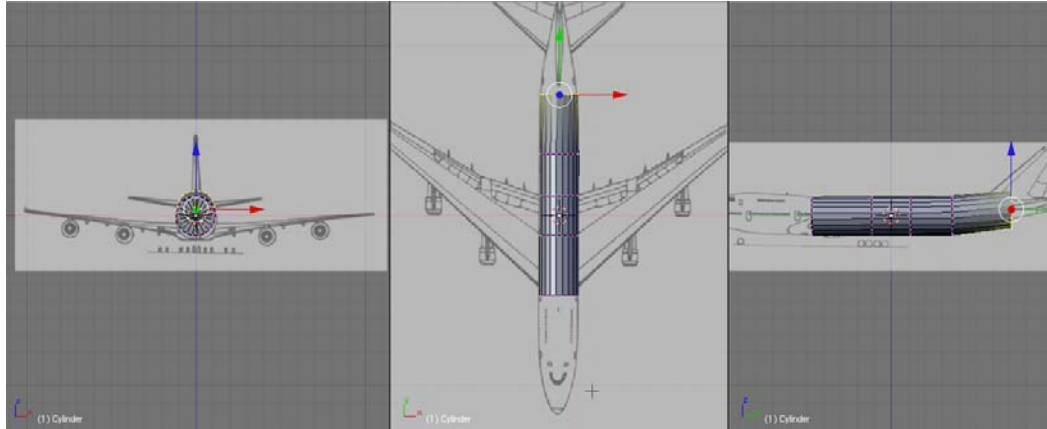


Figure 38. Blender interface with Different views of the Boeing 747.

Using the 3D tools in Blender, such as extrude, scale, rotate, and move the author modeled the object, and finalized it, as shown in Figure 36. It is recommended that when building the model, the author starts with a basic shape such as cylinder, square, or sphere with minimal number of vertices to avoid manipulating complex meshes. Figure 39 shows the model of the airplane with a minimal number of vertices.

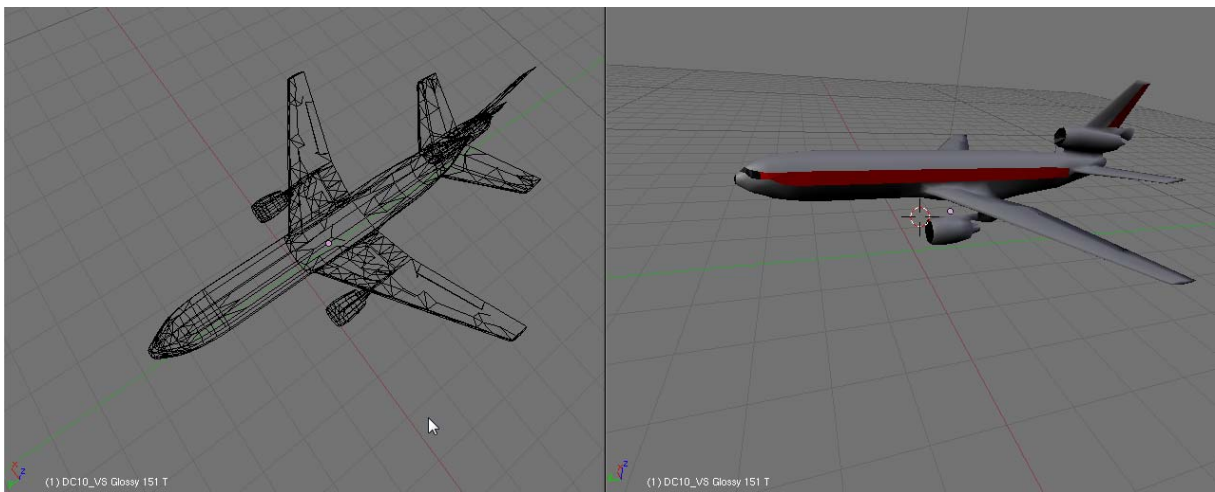


Figure 38. Blender interface showing the model of the airplane with minimal vertices.

Once the model of the airplane is created, Blender is meant to smooth the object being modeled. The starting mesh should not have a lot of vertices, or else the object will

have many rough areas. The “auto-smooth” command in Blender is meant to augment the number of vertices after the object is modeled. An export to X3D procedure is the next step. Figure 40 shows the model of the airplane after smooth operation and ready to be exported in X3D format. Additional instructions about exporting modeled objects from Blender to X3D-Edit are available at:

<https://savage.nps.edu/X3D-Edit/BlenderExportToX3d.html> (Accessed march 2008).

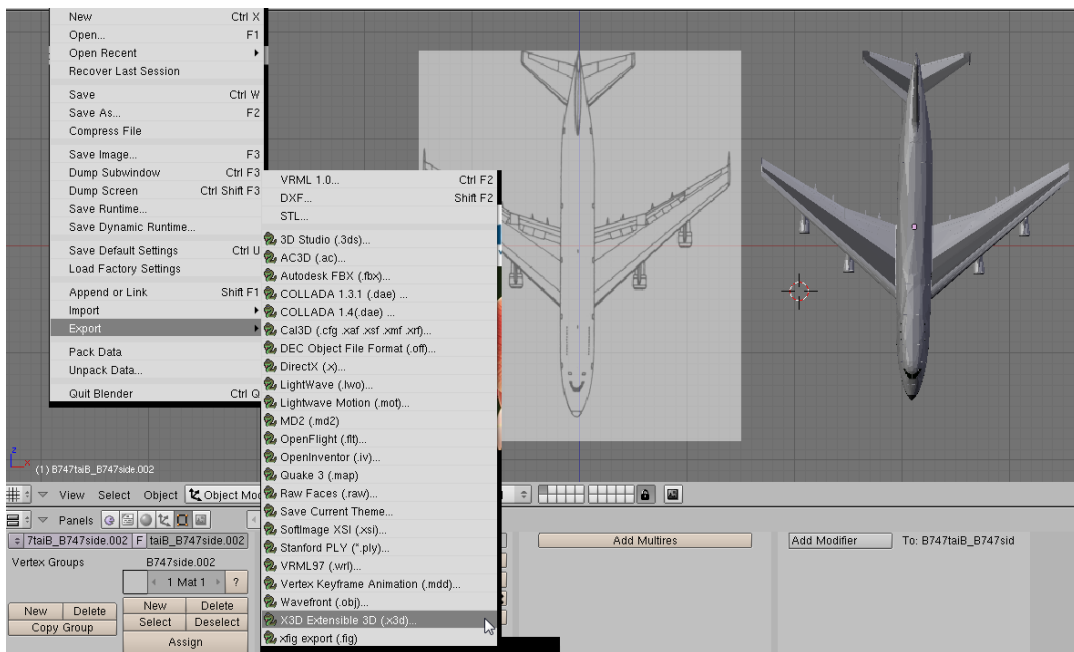


Figure 40. Blender interface with the export feature to X3D model.

Figure 39 illustrates the procedure of exporting the model into X3D format. It is recommended to export the whole model as grouped objects to avoid the manual scaling and the transform operation for each individual sub-object.

After exporting the Blender model to X3D format, extra configurations to the model must be done in order to make it correct . When exporting from Blender to X3D format some configurations are incoherent with the syntax of some node attributes, such as *False* must be changed to *false*. Figure 41 lists some changes to implement in the exported model to X3D format to avoid having errors when visualizing the model in a 3D browser. Further details appear in the following subsections.

| IndexedFaceSet    |   |
|-------------------|---|
| Attribute         | Value   |
| DEF               | <None>  |
| USE               | <None>  |
| coordIndex        | 4 5 6 -1, 4 6 7 -1, 3 4 7 -1, 2 3 7 -1, 1 2 7 -1, 0 1 7 -1, 12 1... |
| ccw               | true <b>1</b>   |
| convex            | false <b>2</b>  |
| solid             | false <b>3</b>  |
| creaseAngle       | 3.14159 <b>4</b>  |
| colorPerVertex    | <None>  |
| colorIndex        | <None>  |
| normalPerVertex   | <None>  |
| normalIndex       | <None>  |
| texCoordIndex     | <None>  |
| set_coordIndex    | <None>  |
| set_colorIndex    | <None>  |
| set_normalIndex   | <None>  |
| set_texCoordIndex | <None>  |
| containerField    | <None>  |
| class             | <None>  |

Figure 41. The four changes to make to the model after exporting from Blender to X3D edit.

**a. Meta Tags**

Additional tags containing information such as name, description, author, translator, created, translated, modified, identifier, reference and license need to be added to the model. X3D-Edit provides good support for this task. Detailed requirements and rationale appears in [Brutzman and Daly 2007], online chapter 15 Metadata Information at

<http://x3dgraphics.com/chapters/Chapter15-MetadataInformation.pdf>

(Accessed march 2008) and the X3D scene Authoring Hints at

<http://www.web3d.org/x3d/content/examples/X3dSceneAuthoringHints.html#metaTags>

(Accessed March 2008).

**b. Scene Corrections**

Some corrections need to be added to the exported X3D model. These problems have been submitted to the Blender community for correction and improvement.

- Boolean values must be in lowercase (e.g. *attribute='TRUE'* becomes *attribute='true'*, similarly *FALSE* becomes *false*).
- The Navigation info node can be removed or commented out.
- The Appearance node following the IndexedFaceSet tag must be updated in order to control the color of the modeled object that is grey by default.

Problem correction regarding Blender export to X3D can be checked via [http://projects.blender.org/tracker/?group\\_id=9&atid=125](http://projects.blender.org/tracker/?group_id=9&atid=125) (Accessed March 2008).

### *c. IndexFaceSet Corrections*

*IndexFaceSet* parameters must be adjusted to fix common export problems:

- To smooth the shading on polygons IndexedFaceSet geometry, set *creaseAngle="3.14159"* .
- To fix non rendering *IndexedFaceSet* geometry from single-sided to double-sided polygons, set *solid="false"*.
- Complex shapes may include concave polygons, which might not render due to default graphics-hardware optimizations, set *convex="false"* to force rendering of all polygons, concave and convex.
- *CCW="true"* might work for some meshes. If possible, go back in Blender and reverse the offending geometry.

The final model of the airplane Boeing 747 after implementing all the changes is shown in Figure 42.

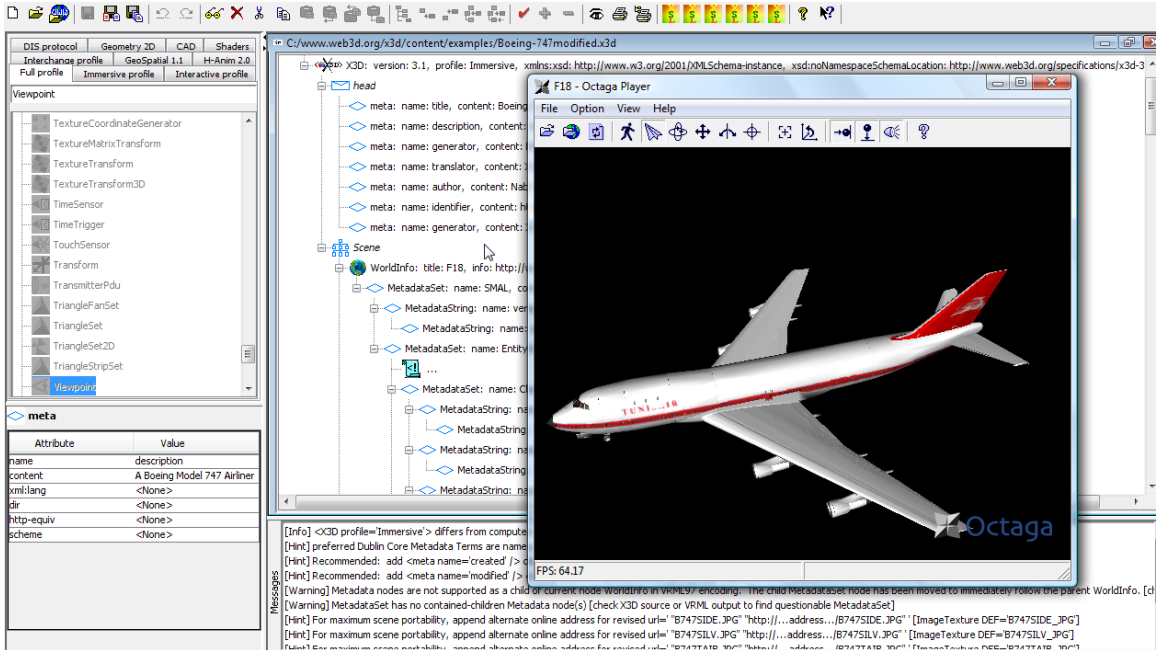


Figure 42. The model of Boeing 747 after making all the required changes on the exported model.

### 3. Model Integration into Master Scene

First, to add the model into the Savage Archive, specific SMAL Metadata must be added to the models, in which a detailed description of the models is given. Then, the models are added to Savage Archive. A catalogue generation tool must be run by the archive administrator. This operation generates icons in the SavageStudio, supporting the ability to drag and drop airplanes on the airport scene. A variety of the catalog information is also produced. In the initial assembly of the full scenario, the airplanes are represented by Simkit entities. To generate the full scene, the airplanes must be represented by the DISMovers3D class. The DISMovers3D have inputs origin coordinates, destination coordinates and speed. The DISMover3D produces DIS network reports that are distributed over the network to animate the moving entities in the scene. Through the DIS pinger, the airplanes are able to follow the specified path characterized by the wait points.



Figure 43. The Tunisia Carthage airport X3D model in Savage Archive view with XJ3D browser.



Figure 44. An X3D scene combining the Boeing 747 model and the Tunisia Carthage airport model. Note proper scale modeling satellite imagery of aircraft next to the terminal.



## **B. SUMMARY**

This chapter has presented the different techniques used to construct the airplane model using Blender, as well as the export procedure to X3D format with minor changes to add to the exported model. It also presented the procedure of constructing the model of Tunisia Carthage airport, starting from elevation and imagery data collection and ending with script Rez to obtain the X3D format of the terrain. The last part in this chapter covered the changes implemented on the terrain model and the airplane model to make them suitable in Savage Archive. Once the models are added and updated, the SavageStudio recognizes these models and their implementation in the master scene become possible.

## **VI. SIMULATION ANALYSIS**

### **A. INTRODUCTION**

The simulation of ground traffic control is meant to give insights of various possible cases that may engender potential accidents. To analyze different possible cases, number of runs of both scenarios is made; data were collected and analyzed through the analysis report generated by Viskit tool as well as through JMP software, which is statistical software for expert data analysis.

### **B. STATISTICS AND ANALYST REPORT**

One of the most valuable features of Viskit tool is the ability to generate an analysis report after each simulation run. This analysis report tracks the state transitions that the author may specify through the property change listener menu when creating the assembly. The analysis report is generated and saved in HTML format and can be displayed from within Viskit. In order to activate the analysis report generation the check box for analysis report must be checked before running the simulation, the author generated two analysis reports, one for each of the two scenarios.

#### **1. Takeoff Scenario**

For the first scenario, which is the takeoff scenario, the number of airplanes is generated by the arrival process. The different mean values of the exponential distributions describing the different time delays are specified. The simulation is run for 100 replications and the simple statistic time-varying PCL is listening to the time spent by the departing airplanes on the taxiways. Since there are not landing airplanes, there are no congestion on the taxiway. The analysis report for the takeoff scenario can be compared with the one generated for the final scenario, in order to test whether or not the simulation is running consistently. The analysis report for the takeoff scenario is in the Appendix A.

## **2. Final Scenario**

The final scenario has three airplanes departing and three airplanes arriving. The goal of the simulation is to test a real situation that requires at least six airplanes trying to get access to the taxiway and runway at the same time. These numbers can be changed to observe the behavior of the simulation. The number of replication is fixed to 100 and the time of the simulation is also fixed to 100 simulation unit times. The generated analysis report is in the Appendix B. One important observation deduct from the two analysis reports is that the time spent by the departing airplanes on the taxiway is higher then the time spent by the landing airplanes on the taxiway. This is due to the landing airplanes having higher priority than the departing airplanes.

## **3. Design of Experiment and Data Collection**

In order to test the simulation of airplane ground traffic, the author chose to vary 4 factors that are related to the traffic on the taxiway. These factors are:

- T 1: Time spent by the departing airplanes to move to runway.
- T 2: Time spent by the departing airplanes to move to taxiway.
- T 3: Time spent by the arriving airplanes to move to taxiway.
- T 4: Time spent by the arriving airplanes to move to arrival gate.

In order to construct design points, the author used Nearly Orthogonal Latin Hypercube (NOLH) script that generates 17 design points by combining the 4 factors specified above. The script of NOLH is available at:

<http://harvest.nps.edu> (Accessed March 2008).

Figure 45 shows the process of generating 17 combinations of the four factors specified above.

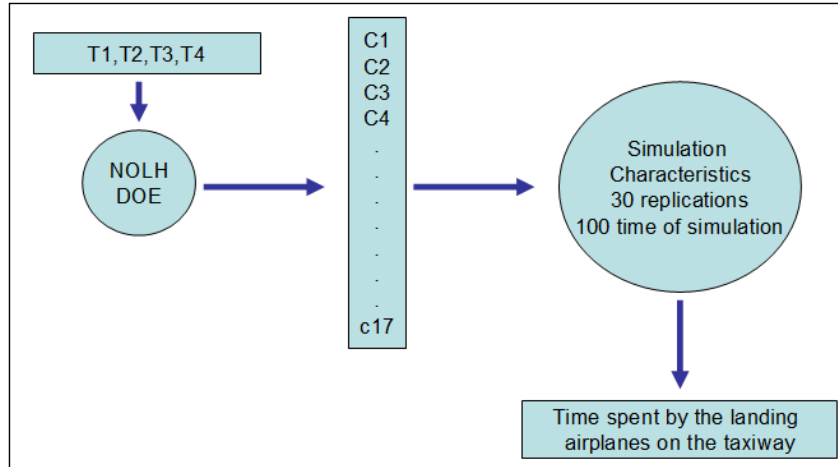


Figure 45. The process of generating the 17 combinations of the four factors with NOLH.

With 30 replications for each combination of the factors, 510 observations were collected and analyzed through JMP software, which is statistical software for expert data analysis. The jump software is mainly used to conduct analysis on data and dynamically links statistics with graphics. Jmp software is available at NPS downloadable software library. <https://www.nps.edu/Technology/SoftwareLib> (Accessed March 2008). The analysis of the distributions of the four factors T1, T2, T3, and T4 as well as the average time spent by the three arriving airplanes on the taxiway is presented in Figure 46.

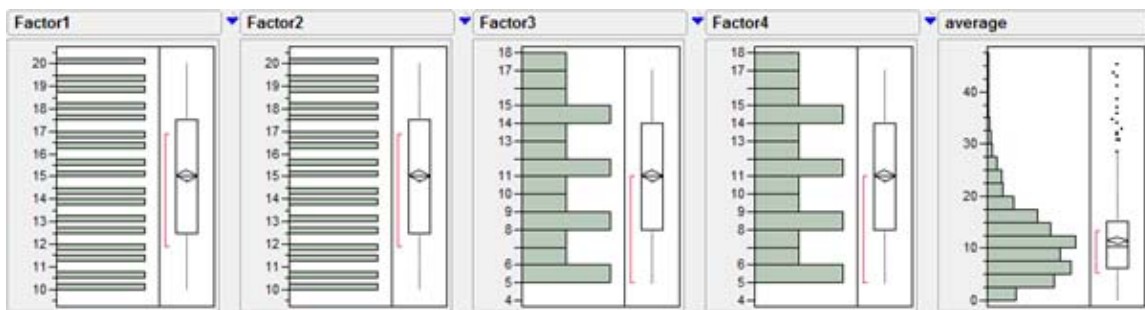


Figure 46. Distribution analysis of T1, T2, T3, and T4 as well as the average time spent by the three arriving airplane on the taxiway.

Figure 47 shows the scatterplot matrix associated to the 510 observations and highlights the concentration of the time spent by the first arriving airplane, the second arriving airplane and the third arriving airplane. The third airplane has the highest time spent on the taxiway while the first airplane has the least time spent on the taxiway.

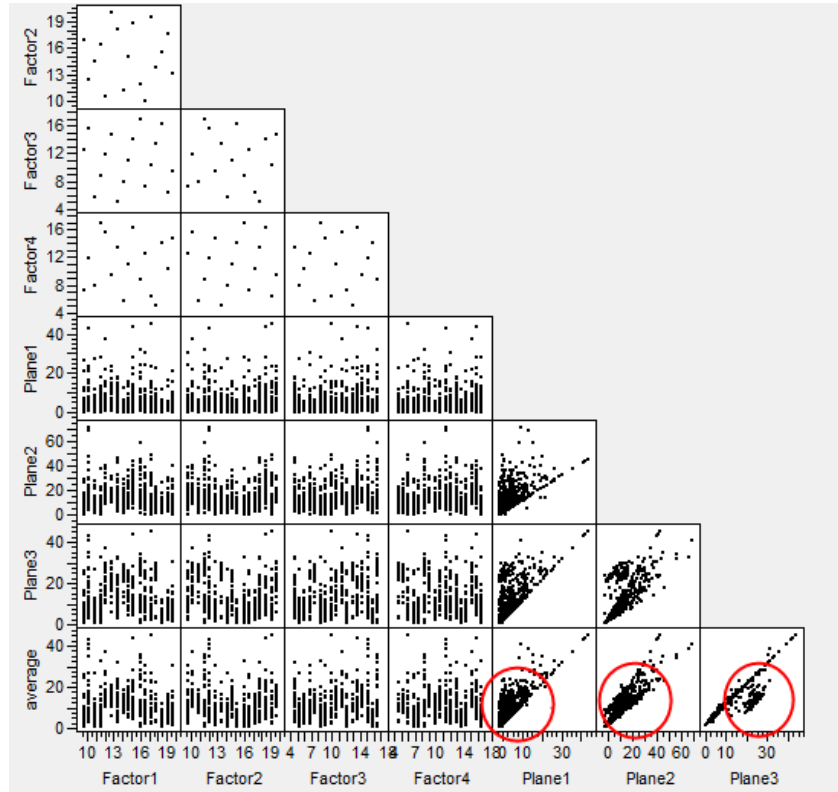


Figure 47. The concentration of the time spent on the taxiway for the first airplane, the second airplane and the third airplane.

The results collected after running the simulation of the full scenario showed the consistency of the model, as well as few limitations. The outliers in Figure 48 illustrates that in some cases the simulation reached the stop time before arriving airplanes accessed the arrival gate. This limitation can be reduced by augmenting the time of simulation by choosing the right configurations before running the simulation.

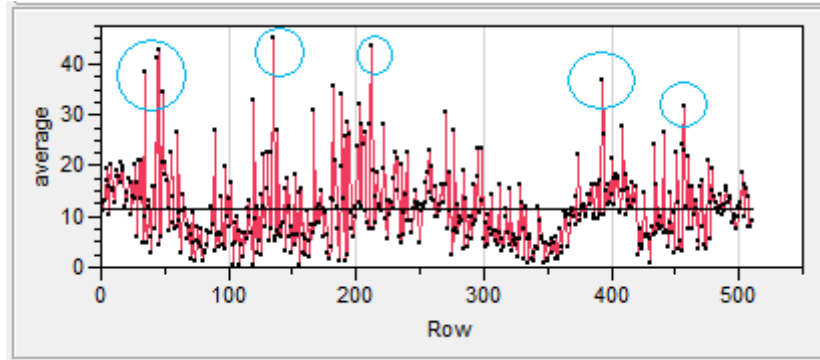


Figure 48. Blue circles showing some outliers for the average of time spent by the arriving airplanes on the taxiway.

### C. SUMMARY

This chapter was dedicated on conducting experiments and collecting data. First, the analysis report generated by Viskit was analyzed. Then a Design Of Experiment was elaborated in order to generate 17 combinations out of four factors by use of Near Orthogonal Latin Hypercube (NOLH) program. Jmp software was used to collect and analyze the outputs of the simulation. The results showed consistency of the model, as well as few limitations. A proper time specification for the model will reduce these limitations, which could be the subject of future work.

THIS PAGE INTENTIONALLY LEFT BLANK

## VII. CONCLUSIONS AND RECOMMENDATIONS

### A. CONCLUSIONS

#### 1. Building the Event Graph

Building the event graph is a complex task, especially when the author had to code the entire simulation without the use of any tool. The Viskit makes the simulation easier to implement. It generates the final code for the simulation and allows the author to debug the code and make eventual changes. Breaking the simulation into different scenarios is a helpful approach. The author broke the scenario into two parts, and followed the same logic in constructing them. For programming complex simulations, starting with a working program that treats the simplest starting situation is an efficient way to begin the simulation coding phase. As the simulation becomes more sophisticated, the author may add additional features by planning their implementation and keeping in a running version of the simulation in each change. This procedure assures the integrity and the consistency of the simulation.

#### 2. Implementing the Assembly

The assembly editor in the Viskit tool offers the possibility to check the simulation coherence, in any time of the coding phase. A detailed description of the assembly editor interface, as well as the implementation of the event graph model is given in Chapter IV. So the student can refer to this thesis to have an initiation to the manipulation of the Viskit tool. Implementing different Property Change Listeners is an efficient way to test the state variables of the simulation.

#### 3. Building the 3D Environment and Models

The physical environment for this thesis is Tunisia Carthage airport. Collecting imagery data and elevation data was not an easy task, since all the public resources have low resolution data for the imagery and sometimes no elevation data at all. Some



commercial resources, such as The NGA Commercial Data Service (NCDS) provided more detailed data for imagery and elevation. NCDS provided better resolution imagery for Tunisia Carthage airport. Global Mapper served to download and store the elevation data. Rez was used to prepare the terrain and imagery data as X3D files. Building the airplane is a time consuming task, but Blender as a graphical 3D tool, is helpful, since it allowed the author to use a blueprint of the airplane as a background image, then using the different tools as extruding scaling and moving simultaneously from different views, makes the construction of the model easier. Exporting the model to X3D format requires additional corrections, such as the attributes of different nodes of the model. Adding the different models to Savage Archive requires adding SMAL Metadata to the model.

## **B. RECOMMENDATIONS FOR FUTURE WORK**

### **1. Scenario Improvement**

The author implemented a very simple scenario of the takeoff and the landing procedures, these scenarios may be extended as a future work by adding different scenarios such as touch and go maneuver, or planning a final mission that includes a takeoff from one airport and landing to another airport. Adding realistic time delays for the airplanes to move from one state to another is also a key for improving the simulation. The author used the same distributions to generate the different time delays between the different states, a future work may take into consideration the test of different distribution such as Gamma distribution or Poisson distribution and investigate which of them represents more realistic behavior. The possibility of adding a specific database to handle the flight plan with a real structure is also an area to which future work may be dedicated.

The author added the feature of fixing the number of departing and landing airplanes to the model. A future work may be allocated to synchronize the time of the simulation with the average of the landing airplanes per day, month, or season. The data used in this thesis is not as realistic as it could be since the time delays are treated

randomly. A future enhancement may be allocated to collect real data of the airplanes taking off and landing and implement these data in the simulation.

## **2. Modeling Constraints**

The collection of the imagery data and the elevation data was not an easy task, since the public servers do not offer detailed data. With the tremendous work being done in X3D Earth in the MOVES institute, saved data may be available for any future work related to generating terrains for specific simulations. Adding extra components to Tunisia Carthage airport such as the tower control can also enhance the visualization of the simulation. For the airplane model, the blueprints make the modeling task easy and less time consuming. The changes made to the exported models can be avoided if the Blender foundation makes adjustment to their export feature. The problems considered in the exported models are the viewpoints adjustment, the alignment of the objects, the attribute values, and the appearance of the models.

## **3. Implementing the Model in SavageStudio:**

Implementing the model in SavageStudio is a beyond the subject of the thesis. Nevertheless, the model Tunisia Carthage airport and of the airplane are already integrated into Savage Archive. Visualizing the simulation is an accomplished task; the remaining operation is to visualize the running simulation with movers instead of airplanes and DISPingers to assure the connection between the assembly and the visualized model. Figure 49 presents a snapshot of SavageStudio tool with the model of the airport terrain and the model of the airplane.

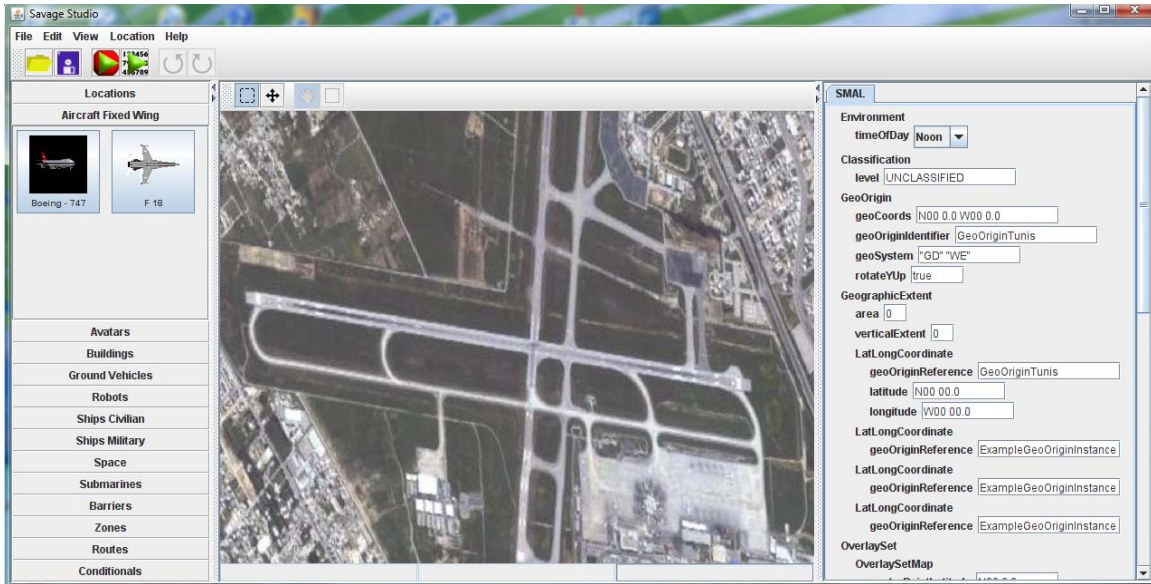


Figure 49. A snapshot of SavageStudio tool illustrating the airport model to the center and the airplane model to the upper right.

After installing airplanes on the terrain of the airport, a configuration of the different components must be done to in the menu shown in the left of Figure 46, such as the max speed of the airplane and the maximum acceleration of the airplane. Once these configurations are done and saved, the scene is generated and viewed in XJ3D browser as illustrated in Figure 50. Completing this capability is an important task for future work.

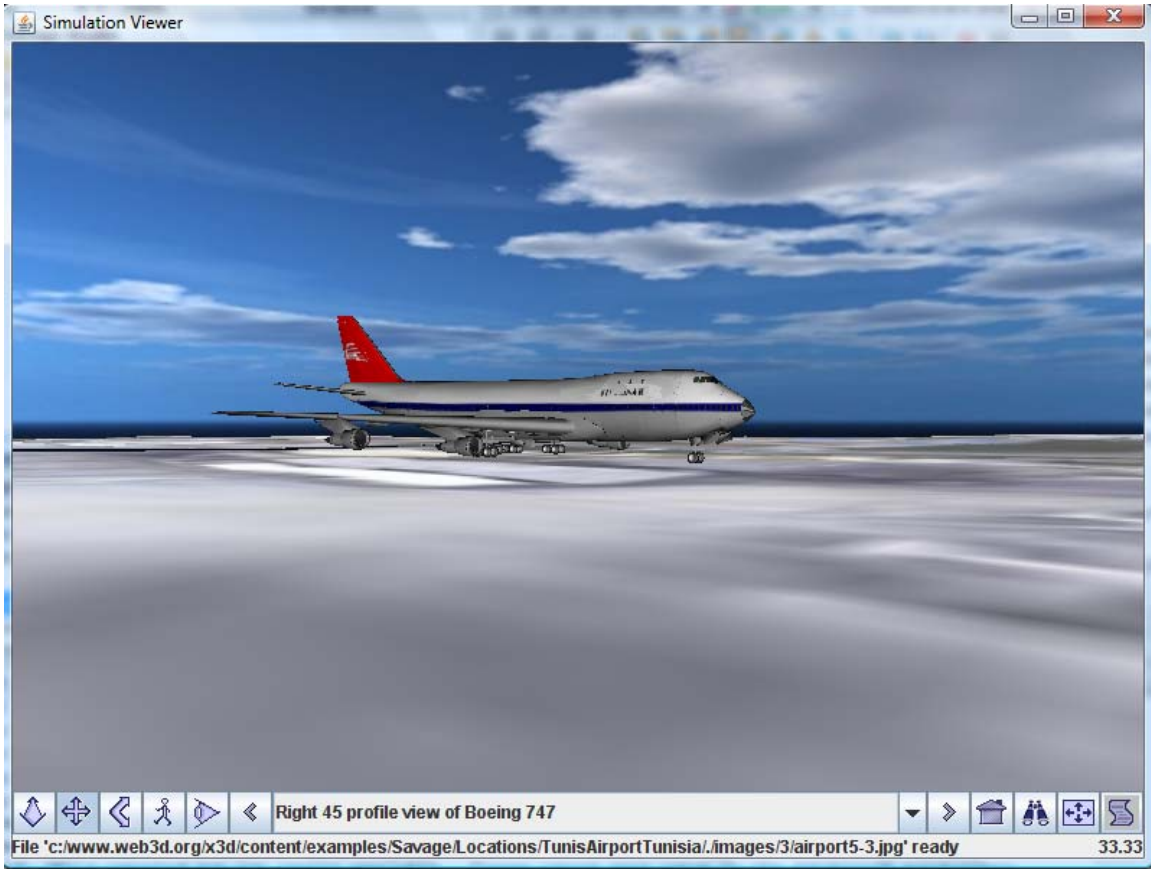


Figure 50. A snapshot of the generated scene via SavageStudio and viewed with XJ3D browser, showing an airplane on the runway.

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX A. ANALYST REPORT FOR THE TAKEOFF SCENARIO

## REPORT FOR THE FIRST SCENARIO THE TAKEOFF SCENARIO

Analyst: **NABIL OUERGI**

Analysis date: **3/27/08 1:20 AM**

[Executive Summary](#) | [Location](#) | [Simulation Configuration](#) | [Entity Parameters](#) | [Behavior Descriptions](#) | [Statistical Results](#) | [Conclusions and Recommendations](#)

---

### **Executive Summary**

#### *Assessment Overview*

This report tracks the time spent by the Departing airplanes on the taxiway without taking into consideration the arrival procedure.

---

### **Simulation Configuration: Viskit Assembly**

The simulation is defined by the Viskit Assembly which collects, lists, initializes and connects all participating entity models within a single scenario. The assembly is then ready for repeated simulation replications, either for visual validation of behavior or statistical analysis of Measures of Effectiveness (MoEs).

#### *Assembly Design Considerations*

The author used a simple stats tally to track the time spent by the departing airplanes on the taxiway

---

### **Summary of Simulation Entities**

| <b>Simulation Entity</b>              | <b>Behavior Definitions</b>                       |
|---------------------------------------|---|
| <a href="#">FirstScenarioModified</a> | <a href="#">scenarioEGs.FirstScenarioModified</a> |

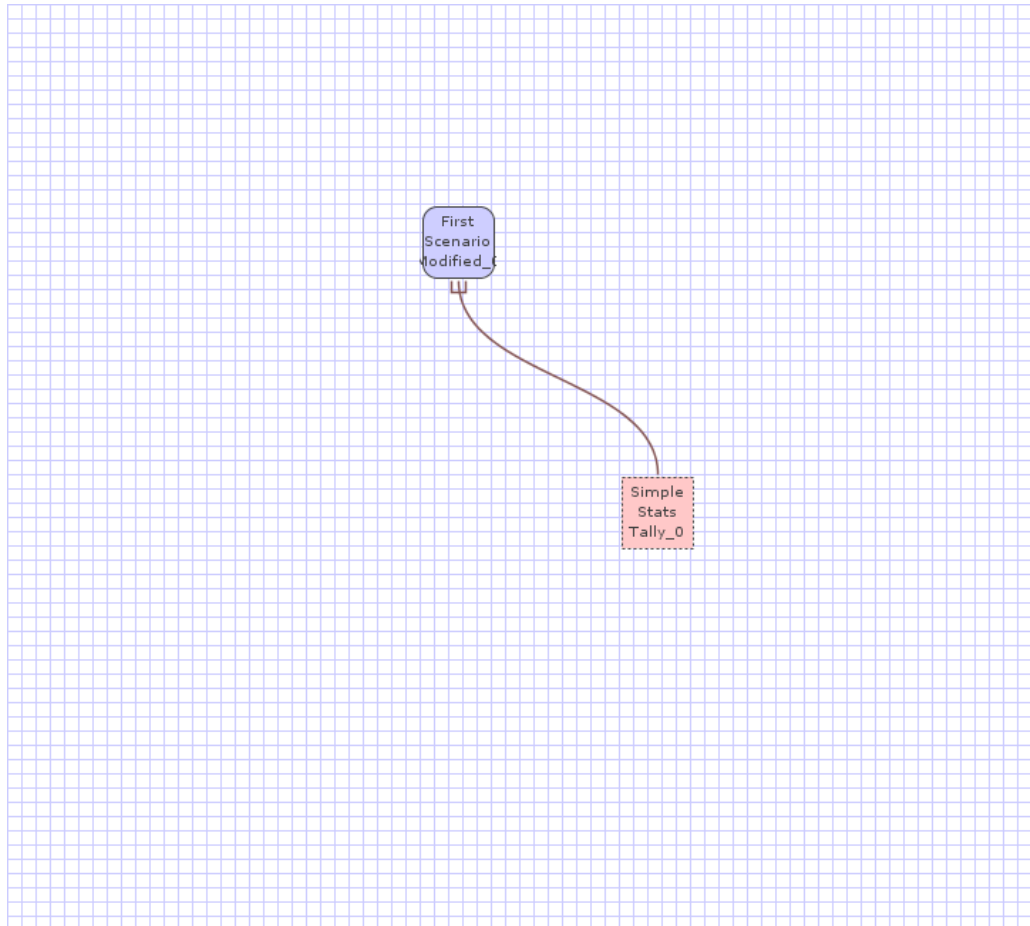


Figure 1: Simulation Assembly Combining all Simulation Entities for this Scenario Experiment

---

### Entity Initialization Parameters for this Simulation Assembly

Initialization parameters are applied to individualize generic behavior models. These parameters customize the event-graph models.

---

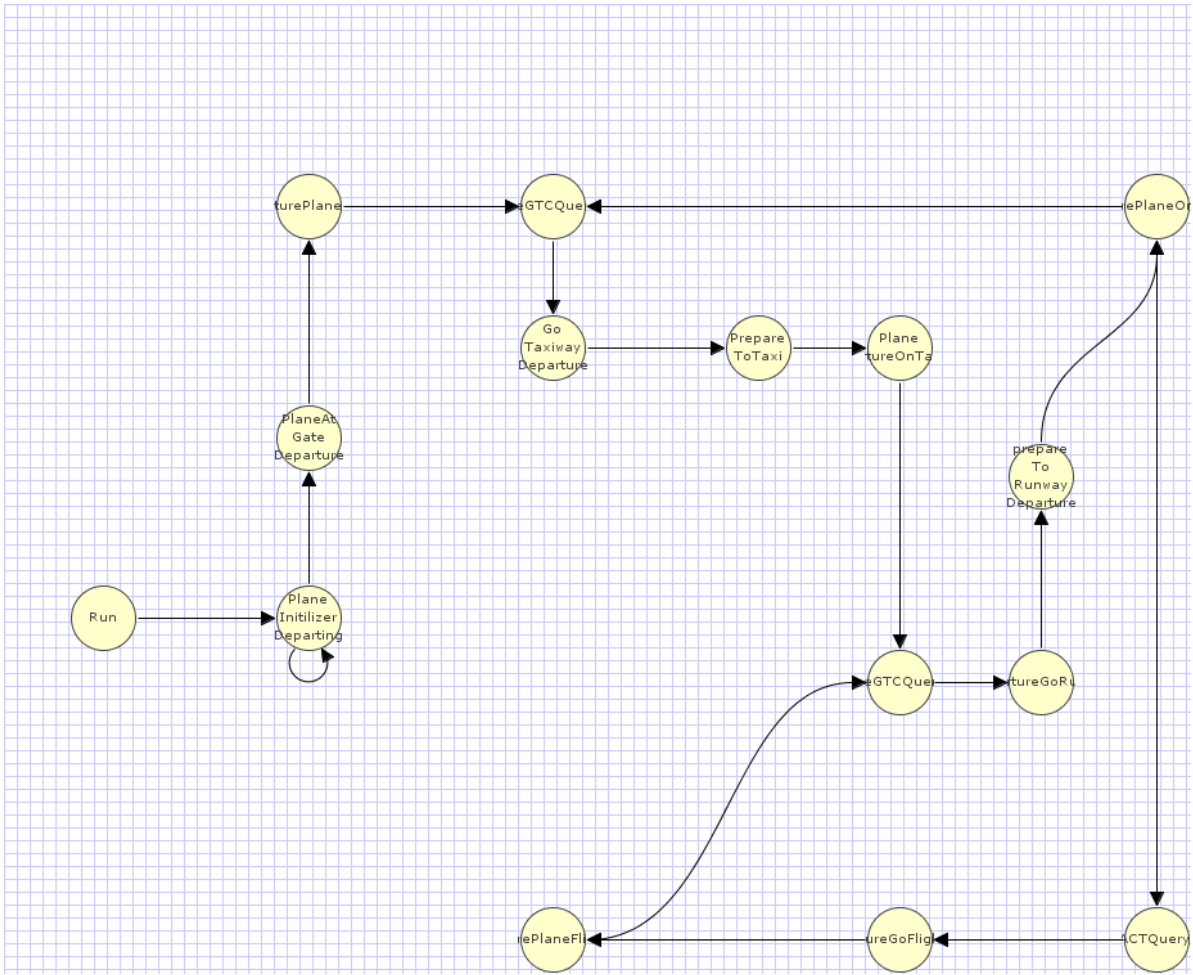


Figure 2: Event Graph for scenarioEGs.FirstScenarioModified

| Initialization Parameter       | Parameter Type              | Description  |
|--------------------------------|-----------------------------|--|
| timeDepartureGate              | simkit.random.RandomVariate | InterArrival time for the plane to come to the gateway |
| timeDepartureReady             | simkit.random.RandomVariate | Time for the plane to get ready to move                |
| timeDepartureTaxiway           | simkit.random.RandomVariate | Time for the plane to go taxiway                       |
| timeDepartureRunway            | simkit.random.RandomVariate | Time for the plane to get to the RunWay                |
| timeDepartureFlighZone         | simkit.random.RandomVariate | Time for the plane to get to the flight zone           |
| initialNumberOfPlanesDeparting | int                         | infinite   |



| State Variable              | Variable Type                       | Description   |
|-----------------------------|-------------------------------------|---|
| queueDepartureGate          | java.util.LinkedList<simkit.Entity> | container for the planes entities at the gate         |
| queueDepartureTaxiway       | java.util.LinkedList<simkit.Entity> | container for the plane at Taxiway                    |
| queueDepartureReady         | java.util.LinkedList<simkit.Entity> | container for the plane at ready to move planes       |
| queueDepartureRunway        | java.util.LinkedList<simkit.Entity> | container for the planes at the runway                |
| queueDepartureFlightZone    | java.util.LinkedList<simkit.Entity> | container for the planes st the flight zone           |
| timeSpentOnTaxiwayDeparture | double                              | the time spent on the Taxiway for the plane departing |

[Back to top](#)

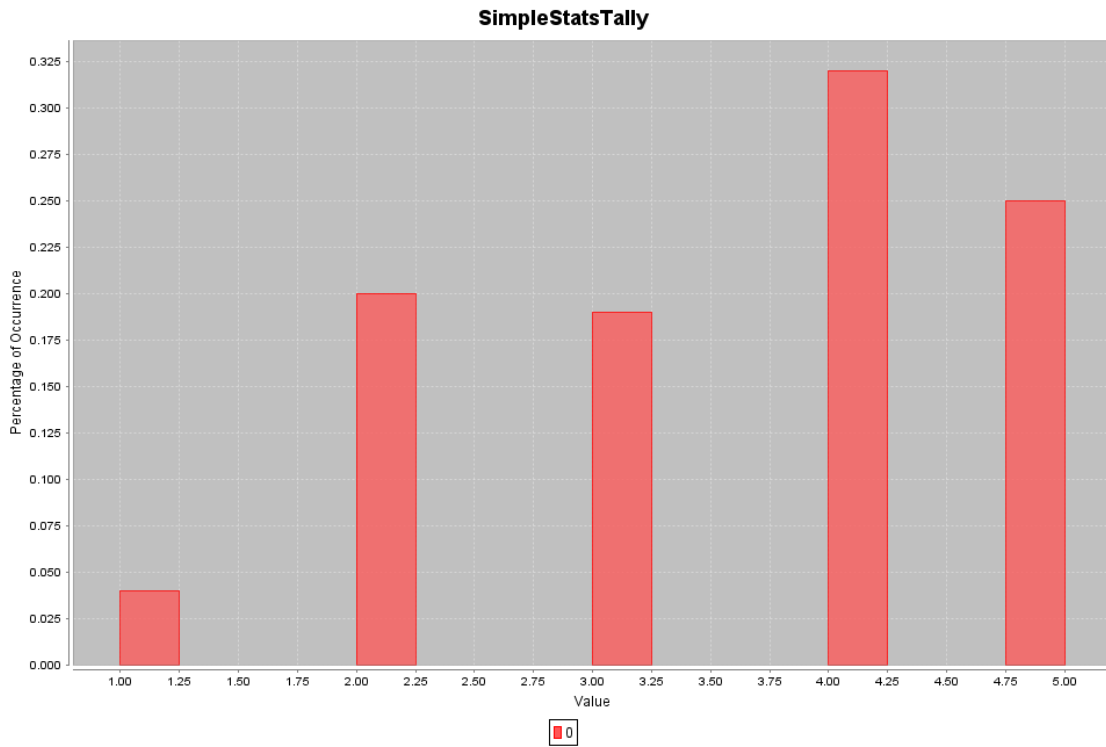


Figure 3: Replications Histogram

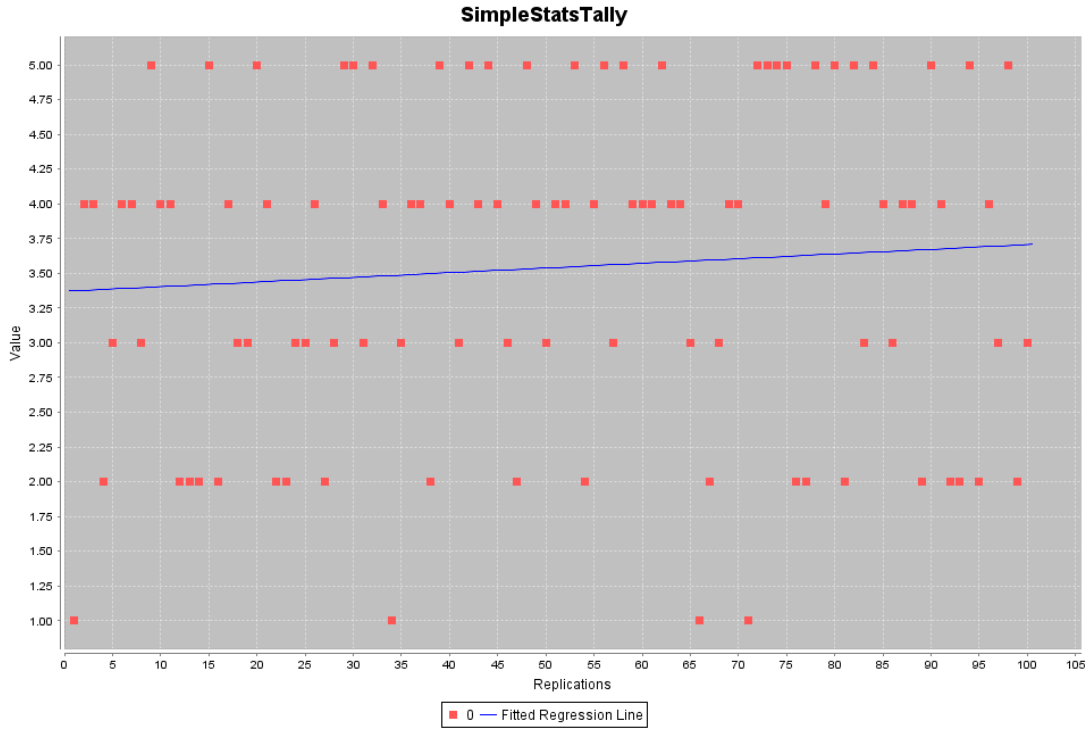


Figure 4: Replications Scatter Plot for 0

| Replication # | Min    | Max    | Mean   | StdDev | Variance |
|---------------|--------|--------|--------|--------|----------|
| 1             | 3.048  | 10.699 | 6.842  | 4.223  | 22.305   |
| 2             | 4.048  | 14.699 | 7.842  | 4.723  | 22.305   |
| 3             | 2.395  | 22.079 | 11.091 | 8.887  | 78.984   |
| 4             | 5.093  | 5.356  | 5.224  | 0.186  | 0.035    |
| 5             | 3.130  | 24.748 | 13.084 | 10.910 | 119.032  |
| 6             | 0.210  | 8.790  | 4.861  | 4.569  | 20.877   |
| 7             | 2.712  | 46.878 | 22.274 | 18.265 | 333.614  |
| 8             | 1.646  | 19.103 | 8.826  | 9.131  | 83.376   |
| 9             | 2.343  | 21.950 | 13.576 | 7.487  | 56.056   |
| 10            | 0.047  | 26.443 | 15.575 | 12.252 | 150.103  |
| 11            | 1.089  | 17.448 | 10.231 | 8.331  | 69.408   |
| 12            | 13.197 | 25.815 | 19.506 | 8.922  | 79.605   |
| 13            | 0.420  | 9.612  | 5.016  | 6.500  | 42.247   |
| 14            | 1.001  | 2.735  | 1.868  | 1.227  | 1.505    |
| 15            | 0.285  | 14.574 | 6.858  | 5.547  | 30.774   |
| 16            | 12.411 | 14.720 | 13.565 | 1.632  | 2.665    |

|    |        |        |        |        |         |
|----|--------|--------|--------|--------|---------|
| 17 | 2.203  | 20.257 | 10.251 | 7.923  | 62.766  |
| 18 | 1.989  | 20.577 | 13.318 | 9.940  | 98.812  |
| 19 | 0.465  | 4.733  | 2.080  | 2.315  | 5.361   |
| 20 | 3.815  | 17.947 | 10.829 | 5.239  | 27.449  |
| 21 | 0.940  | 12.312 | 5.699  | 5.038  | 25.381  |
| 22 | 21.505 | 22.399 | 21.952 | 0.632  | 0.399   |
| 23 | 2.498  | 36.161 | 19.330 | 23.803 | 566.601 |
| 24 | 2.365  | 8.969  | 5.710  | 3.303  | 10.909  |
| 25 | 13.720 | 26.415 | 18.118 | 7.189  | 51.685  |
| 26 | 7.194  | 19.566 | 12.615 | 6.013  | 36.158  |
| 27 | 1.280  | 7.667  | 4.473  | 4.517  | 20.402  |
| 28 | 1.721  | 24.502 | 10.629 | 12.175 | 148.234 |
| 29 | 3.521  | 20.467 | 14.560 | 6.830  | 46.649  |
| 30 | 1.682  | 10.030 | 5.741  | 3.430  | 11.765  |
| 31 | 0.028  | 18.575 | 6.343  | 10.595 | 112.264 |
| 32 | 4.366  | 22.258 | 10.847 | 6.950  | 48.305  |
| 33 | 0.610  | 13.665 | 7.646  | 6.899  | 47.592  |
| 34 | 2.274  | 15.565 | 8.337  | 6.722  | 45.181  |
| 35 | 1.472  | 9.805  | 4.551  | 4.572  | 20.907  |
| 36 | 2.191  | 28.255 | 16.285 | 11.352 | 128.874 |
| 37 | 5.535  | 22.776 | 16.379 | 8.122  | 65.965  |
| 38 | 0.138  | 17.830 | 8.984  | 12.510 | 156.505 |
| 39 | 7.395  | 42.452 | 20.637 | 14.868 | 221.046 |
| 40 | 0.695  | 9.132  | 5.461  | 3.507  | 12.301  |
| 41 | 7.776  | 45.885 | 21.238 | 21.376 | 456.927 |
| 42 | 7.676  | 12.778 | 9.980  | 1.909  | 3.646   |
| 43 | 4.072  | 7.607  | 6.131  | 1.488  | 2.215   |
| 44 | 1.524  | 16.933 | 11.118 | 5.791  | 33.535  |
| 45 | 0.896  | 19.715 | 7.537  | 8.323  | 69.276  |
| 46 | 10.070 | 24.425 | 15.051 | 8.123  | 65.990  |
| 47 | 0.024  | 3.424  | 1.724  | 2.404  | 5.780   |
| 48 | 1.922  | 10.488 | 6.491  | 3.063  | 9.382   |
| 49 | 0.642  | 29.994 | 10.670 | 13.759 | 189.303 |
| 50 | 3.402  | 11.510 | 8.287  | 4.302  | 18.507  |

|    |        |        |        |        |         |
|----|--------|--------|--------|--------|---------|
| 51 | 5.535  | 15.506 | 10.944 | 4.354  | 18.960  |
| 52 | 1.684  | 20.289 | 11.017 | 8.838  | 78.105  |
| 53 | 14.154 | 31.569 | 21.623 | 6.479  | 41.976  |
| 54 | 2.874  | 15.562 | 9.218  | 8.971  | 80.487  |
| 55 | 0.793  | 16.271 | 9.117  | 6.355  | 40.382  |
| 56 | 5.832  | 20.103 | 12.959 | 5.158  | 26.604  |
| 57 | 8.491  | 48.163 | 23.728 | 21.376 | 456.918 |
| 58 | 6.929  | 23.163 | 15.750 | 7.002  | 49.034  |
| 59 | 0.144  | 6.588  | 2.684  | 3.026  | 9.158   |
| 60 | 7.455  | 13.711 | 9.658  | 2.827  | 7.994   |
| 61 | 7.494  | 14.346 | 9.535  | 3.240  | 10.498  |
| 62 | 1.429  | 26.944 | 12.756 | 9.179  | 84.246  |
| 63 | 1.056  | 4.767  | 2.888  | 1.516  | 2.298   |
| 64 | 6.835  | 28.089 | 13.163 | 10.038 | 100.764 |
| 65 | 8.436  | 13.743 | 10.617 | 2.776  | 7.708   |
| 66 | 2.274  | 15.565 | 8.337  | 6.722  | 45.181  |
| 67 | 5.994  | 19.521 | 12.757 | 9.565  | 91.491  |
| 68 | 0.062  | 15.755 | 7.950  | 7.846  | 61.567  |
| 69 | 8.591  | 26.008 | 14.803 | 8.196  | 67.180  |
| 70 | 4.510  | 18.481 | 12.153 | 6.290  | 39.559  |
| 71 | 2.274  | 15.565 | 8.337  | 6.722  | 45.181  |
| 72 | 1.225  | 13.498 | 6.812  | 5.072  | 25.721  |
| 73 | 11.770 | 20.980 | 15.865 | 3.636  | 13.222  |
| 74 | 3.332  | 58.365 | 24.077 | 21.908 | 479.974 |
| 75 | 0.578  | 12.955 | 4.270  | 4.977  | 24.770  |
| 76 | 6.190  | 9.275  | 7.733  | 2.181  | 4.757   |
| 77 | 6.070  | 11.061 | 8.565  | 3.529  | 12.453  |
| 78 | 0.837  | 15.549 | 7.594  | 5.606  | 31.424  |
| 79 | 4.097  | 21.877 | 11.813 | 7.948  | 63.173  |
| 80 | 4.021  | 12.919 | 8.705  | 3.725  | 13.878  |
| 81 | 12.488 | 22.743 | 17.615 | 7.251  | 52.579  |
| 82 | 3.304  | 27.207 | 12.839 | 9.208  | 84.790  |
| 83 | 2.274  | 15.565 | 8.337  | 6.722  | 45.181  |
| 84 | 1.293  | 24.971 | 12.120 | 10.151 | 103.052 |

|     |       |        |        |        |         |
|-----|-------|--------|--------|--------|---------|
| 85  | 0.267 | 24.286 | 15.443 | 10.477 | 109.774 |
| 86  | 5.801 | 12.652 | 9.770  | 3.553  | 12.625  |
| 87  | 8.292 | 36.554 | 19.401 | 13.141 | 172.697 |
| 88  | 7.091 | 17.174 | 10.317 | 4.650  | 21.620  |
| 89  | 0.355 | 23.053 | 11.704 | 16.050 | 257.592 |
| 90  | 3.488 | 13.698 | 6.148  | 4.252  | 18.079  |
| 91  | 1.112 | 6.721  | 3.614  | 2.676  | 7.162   |
| 92  | 0.452 | 6.250  | 3.351  | 4.100  | 16.807  |
| 93  | 0.894 | 30.843 | 15.869 | 21.177 | 448.450 |
| 94  | 2.595 | 18.770 | 8.468  | 6.223  | 38.729  |
| 95  | 1.038 | 10.214 | 5.626  | 6.488  | 42.100  |
| 96  | 1.978 | 34.873 | 17.284 | 13.561 | 183.899 |
| 97  | 5.207 | 19.734 | 10.683 | 7.896  | 62.351  |
| 98  | 1.388 | 10.252 | 4.805  | 3.848  | 14.806  |
| 99  | 0.434 | 25.712 | 13.073 | 17.875 | 319.502 |
| 100 | 4.460 | 17.842 | 11.377 | 6.702  | 44.921  |

[Back to top](#)

## Summary Report

| Entity           | MoE / MoP | # Replications | Min   | Max   | Mean  | StdDev | Variance |
|------------------|-----------|----------------|-------|-------|-------|--------|----------|
| SimpleStatsTally | 0         | 100            | 1.000 | 5.000 | 3.540 | 1.184  | 1.402    |

[Back to top](#)

---

## Conclusions and Recommendations

### *Conclusions*

The author used one type of distribution (Exponential distribution) with different means to generate the time that takes the planes to move from one location to another

### *Recommendations for Future Work*

A further investigation on real data may present a subject for future work

[Back to top](#)

---

This report was autogenerated by the Viskit Event Graph and Assembly modeling tool using Simkit discrete-event simulation (DES) libraries. Online at <https://diana.nps.edu/Viskit> and <https://diana.nps.edu/Simkit>.

## APPENDIX B. ANALYST REPORT FOR THE FINAL SCENARIO TAKEOFF AND LANDING PROCEDURES

### REPORT ANALYSIS FOR THE FINAL SCENARIO ARRIVING AND DEPARTING PLANES

Analyst: **NABIL OUERGHI**

Analysis date: **3/27/08 12:57 AM**

[Executive Summary](#) | [Location](#) | [Simulation Configuration](#) | [Entity Parameters](#) | [Behavior Descriptions](#) | [Statistical Results](#) | [Conclusions and Recommendations](#)

---

#### **Executive Summary**

##### *Assessment Overview*

\*\*\*This analysis report tracks the time spent by the arriving airplanes on the taxiway\*\*\*

[Back to top](#)

---

#### **Location of Simulation**

##### *Description of Location Features*

\*\*\*The simulation location is Tunisia Carthage Airport\*\*\*

##### *Post-Experiment Analysis of Significant Location Features*

\*\*\*The author specified the different means of the exponential distribution allocated to the times that take plane to move from a location to the other\*\*\*

---

#### **Simulation Configuration: Viskit Assembly**

The simulation is defined by the Viskit Assembly which collects, lists, initializes and connects all participating entity models within a single scenario. The assembly is then ready for repeated simulation replications, either for visual validation of behavior or statistical analysis of Measures of Effectiveness (MoEs).

##### *Assembly Design Considerations*

\*\*\*Since the state variable tracked in this report is the time spent by the arriving airplanes on the taxiway, the author chosed to use a simple stats tally property listener\*\*\*

##### *Production Notes*

All units are meters and degrees unless otherwise noted.

##### *Post-Experiment Analysis of Simulation Assembly Design*

Test whether or not the arriving airplanes spend less time on the taxiway than the departing planes

---

#### **Summary of Simulation Entities**

| <b>Simulation Entity</b> | <b>Behavior Definitions</b>               |
|--------------------------|---|
| ScenarioWhole            | <a href="#">scenarioEGs.ScenarioWhole</a> |

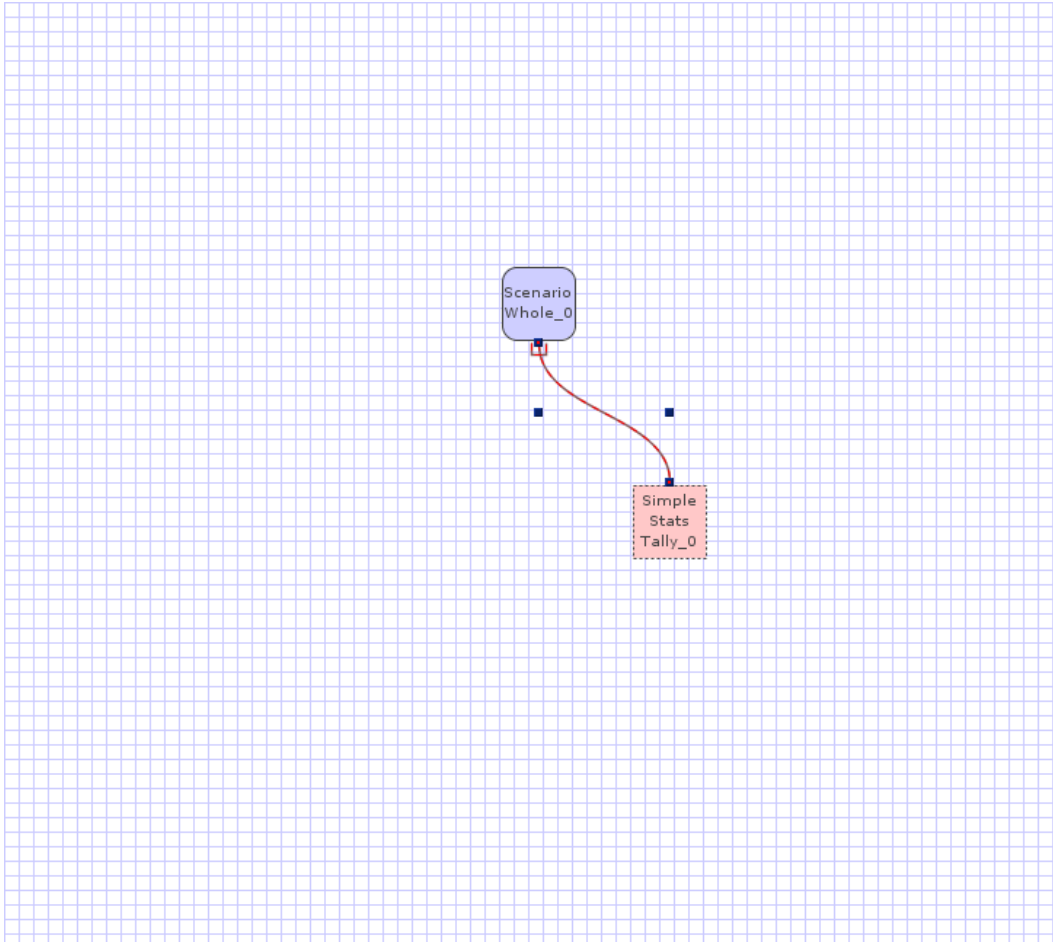


Figure 1: Simulation Assembly Combining all Simulation Entities for this Scenario Experiment

---

### **Entity Initialization Parameters for this Simulation Assembly**

Initialization parameters are applied to individualize generic behavior models. These parameters customize the event-graph models.

---

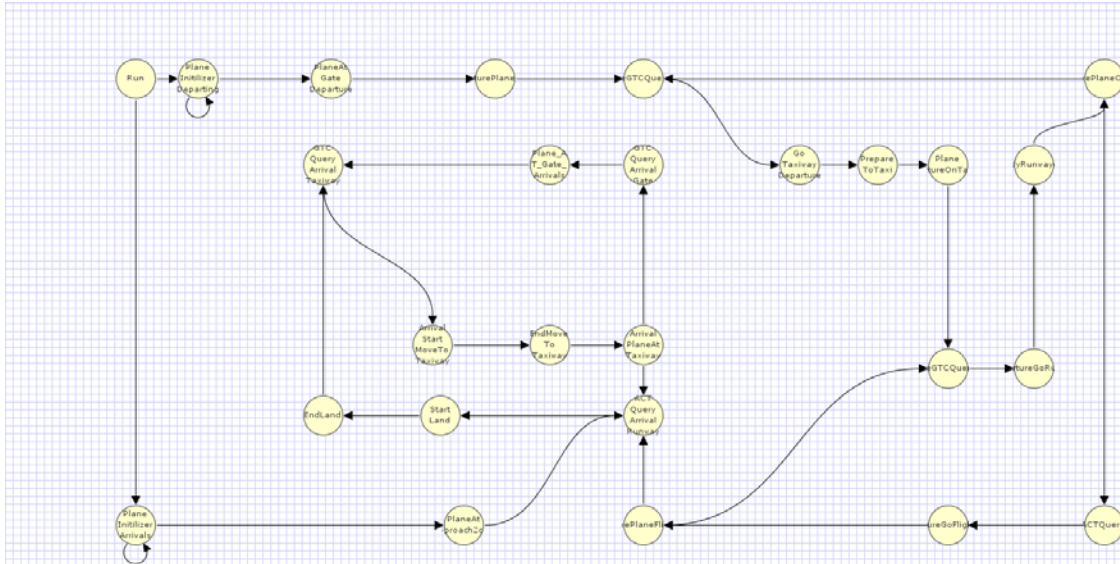


Figure 2: Event Graph for scenarioEGs.ScenarioWhole

| Initialization Parameter | Parameter Type              | Description  |
|--------------------------|-----------------------------|--|
| timeDepartureGate        | simkit.random.RandomVariate | InterArrival time for the plane to come to the gateway |
| timeDepartureReady       | simkit.random.RandomVariate | Time for the plane to get ready to move                |
| timeDepartureTaxiway     | simkit.random.RandomVariate | Time for the plane to go taxiway                       |
| timeDepartureRunway      | simkit.random.RandomVariate | Time for the plane to get to the RunWay                |
| timeDepartureFlighZone   | simkit.random.RandomVariate | Time for the plane to get to the flight zone           |
| timeArrivalApproachZone  | simkit.random.RandomVariate | Time for plane to come to the Approach zone            |
| timeArrivalRunway        | simkit.random.RandomVariate | Time that takes the plane to land                      |
| timeArrivalTaxiway       | simkit.random.RandomVariate | Time that takes the plane to go from Runway to Taxiway |
| timeArrivalGate          | simkit.random.RandomVariate | Time that takes the plane to go to the Arrival gate    |



| initialNumberOfPlanesDeparting | int                                 | 3   |
|--------------------------------|-------------------------------------|---|
| initialNumberOfPlanesArriving  | int                                 | 3   |
| State Variable                 | Variable Type                       | Description   |
| queueDepartureGate             | java.util.LinkedList<simkit.Entity> | container for the planes entities at the gate             |
| queueDepartureTaxiway          | java.util.LinkedList<simkit.Entity> | container for the plane at Taxiway                        |
| queueDepartureReady            | java.util.LinkedList<simkit.Entity> | container for the plane at ready to move planes           |
| queueDepartureRunway           | java.util.LinkedList<simkit.Entity> | container for the planes at the runway                    |
| queueDepartureFlightZone       | java.util.LinkedList<simkit.Entity> | container for the planes at the flight zone               |
| queueArrivalApproachZone       | java.util.LinkedList<simkit.Entity> | Container for the planes arriving to the Flight Zone      |
| queueArrivalRunway             | java.util.LinkedList<simkit.Entity> | Container for the plane landing                           |
| queueArrivalTaxiway            | java.util.LinkedList<simkit.Entity> | Container for the plane landing to taxiway                |
| queueArrivalGate               | java.util.LinkedList<simkit.Entity> | Container for the planes landing and coming to the gate   |
| timeSpentOnTaxiwayArrival      | double                              | Measures the time spent on the taxiway for plane arriving |
| timeSpentOnTaxiwayDeparture    | double                              | the time spent on the Taxiway for the plane departing     |

[Back to top](#)

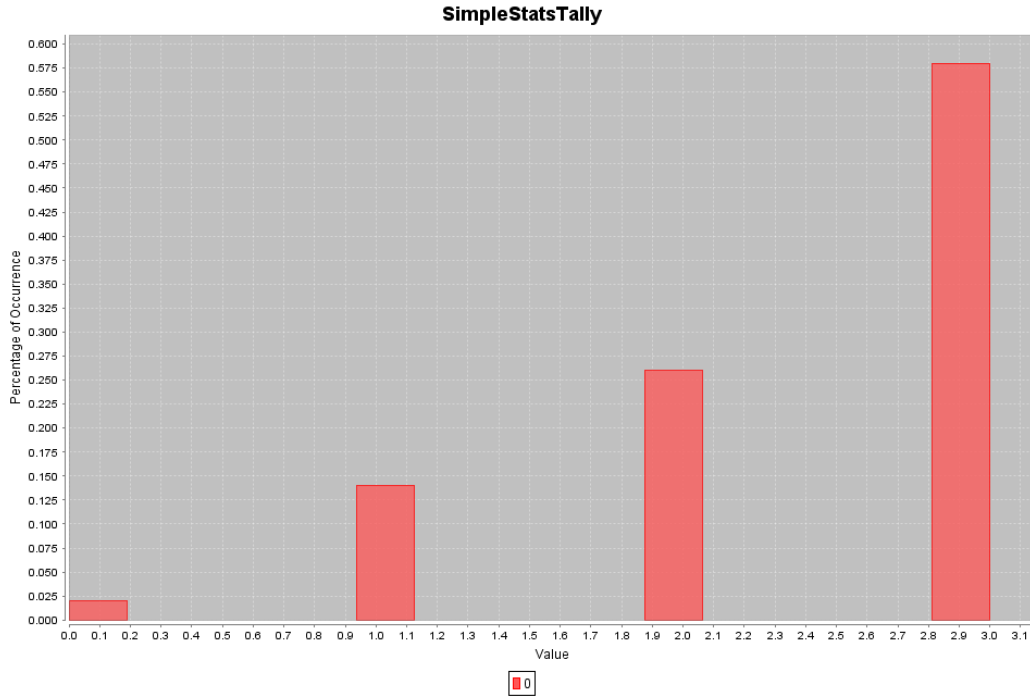


Figure 3: Replications Histogram  
**SimpleStatsTally**

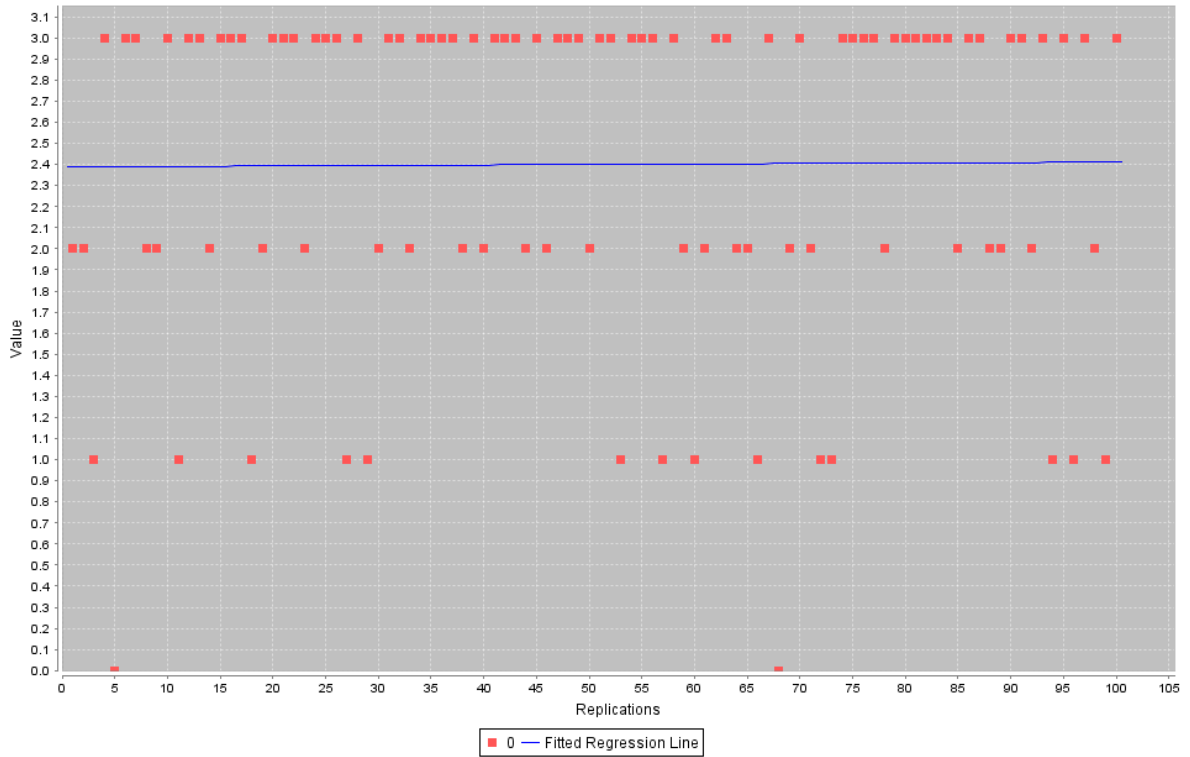


Figure 4: Replications Scatter Plot

| Replication # | Min    | Max    | Mean   | StdDev | Variance |
|---------------|--------|--------|--------|--------|----------|
| 1             | 4.739  | 8.492  | 6.615  | 2.654  | 7.041    |
| 2             | 11.307 | 19.725 | 15.516 | 5.953  | 35.436   |
| 3             | 7.317  | 9.735  | 14.412 | 5.853  | 33.436   |
| 4             | 6.453  | 18.091 | 12.579 | 5.843  | 34.145   |
| 5             | 6.632  | 19.066 | 14.114 | 6.591  | 43.447   |
| 6             | 1.583  | 14.315 | 6.271  | 6.998  | 48.970   |
| 7             | 2.834  | 13.256 | 7.462  | 5.308  | 28.176   |
| 8             | 4.856  | 5.121  | 4.989  | 0.187  | 0.035    |
| 9             | 0.289  | 6.127  | 3.208  | 4.128  | 17.038   |
| 10            | 0.516  | 1.625  | 0.969  | 0.582  | 0.339    |
| 11            | 4.505  | 10.285 | 7.428  | 2.891  | 8.357    |
| 12            | 0.318  | 6.452  | 3.315  | 3.070  | 9.424    |
| 13            | 6.632  | 19.066 | 14.114 | 6.591  | 43.447   |
| 14            | 1.254  | 12.300 | 6.777  | 7.811  | 61.011   |
| 15            | 1.997  | 19.063 | 7.943  | 9.638  | 92.891   |
| 16            | 1.649  | 11.167 | 5.156  | 5.230  | 27.348   |
| 17            | 2.962  | 23.382 | 11.426 | 10.648 | 113.386  |
| 18            | 4.505  | 10.285 | 7.428  | 2.891  | 8.357    |
| 19            | 3.487  | 16.118 | 9.803  | 8.932  | 79.777   |
| 20            | 4.088  | 14.201 | 7.882  | 5.509  | 30.351   |
| 21            | 2.161  | 15.525 | 8.215  | 6.770  | 45.832   |
| 22            | 6.041  | 43.129 | 30.435 | 21.132 | 446.564  |
| 23            | 14.490 | 28.843 | 21.667 | 10.149 | 103.008  |
| 24            | 11.432 | 27.114 | 20.706 | 8.225  | 67.647   |
| 25            | 0.233  | 19.499 | 10.752 | 9.754  | 95.150   |
| 26            | 14.804 | 29.567 | 22.097 | 7.383  | 54.514   |
| 27            | 4.505  | 10.285 | 7.428  | 2.891  | 8.357    |
| 28            | 2.537  | 21.345 | 12.555 | 9.464  | 89.566   |
| 29            | 4.505  | 10.285 | 7.428  | 2.891  | 8.357    |
| 30            | 0.027  | 10.943 | 5.485  | 7.719  | 59.583   |
| 31            | 1.097  | 7.918  | 4.670  | 3.422  | 11.710   |
| 32            | 0.964  | 8.342  | 5.212  | 3.814  | 14.547   |

|    |        |        |        |        |         |
|----|--------|--------|--------|--------|---------|
| 33 | 14.586 | 24.826 | 19.706 | 7.241  | 52.431  |
| 34 | 0.045  | 10.831 | 6.304  | 5.598  | 31.332  |
| 35 | 7.684  | 45.828 | 29.067 | 19.488 | 379.777 |
| 36 | 11.117 | 16.313 | 14.038 | 2.658  | 7.065   |
| 37 | 2.354  | 8.064  | 5.826  | 3.049  | 9.293   |
| 38 | 5.106  | 24.366 | 14.736 | 13.619 | 185.485 |
| 39 | 0.293  | 10.177 | 3.798  | 5.534  | 30.624  |
| 40 | 4.587  | 12.749 | 8.668  | 5.771  | 33.304  |
| 41 | 2.197  | 7.196  | 4.858  | 2.515  | 6.326   |
| 42 | 2.492  | 20.094 | 11.268 | 8.801  | 77.459  |
| 43 | 2.140  | 5.875  | 3.720  | 1.933  | 3.736   |
| 44 | 5.303  | 14.607 | 9.955  | 6.579  | 43.281  |
| 45 | 14.078 | 27.710 | 19.042 | 7.533  | 56.751  |
| 46 | 6.687  | 27.581 | 17.134 | 14.775 | 218.290 |
| 47 | 0.104  | 22.198 | 7.971  | 12.344 | 152.376 |
| 48 | 1.530  | 15.815 | 7.960  | 7.249  | 52.544  |
| 49 | 4.842  | 17.445 | 10.045 | 6.583  | 43.331  |
| 50 | 2.835  | 18.072 | 10.454 | 10.774 | 116.076 |
| 51 | 7.900  | 16.749 | 12.980 | 4.568  | 20.867  |
| 52 | 2.759  | 8.323  | 4.811  | 3.056  | 9.337   |
| 53 | 4.505  | 10.285 | 7.428  | 2.891  | 8.357   |
| 54 | 0.653  | 11.510 | 4.680  | 5.947  | 35.364  |
| 55 | 0.679  | 12.834 | 7.964  | 6.428  | 41.315  |
| 56 | 0.643  | 8.472  | 3.267  | 4.508  | 20.319  |
| 57 | 4.505  | 10.285 | 7.428  | 2.891  | 8.357   |
| 58 | 7.828  | 11.386 | 9.164  | 1.937  | 3.751   |
| 59 | 13.848 | 22.444 | 18.146 | 6.078  | 36.942  |
| 60 | 4.505  | 10.285 | 7.428  | 2.891  | 8.357   |
| 61 | 11.249 | 12.742 | 11.996 | 1.056  | 1.114   |
| 62 | 2.507  | 15.692 | 7.951  | 6.886  | 47.413  |
| 63 | 1.741  | 7.646  | 4.611  | 2.956  | 8.737   |
| 64 | 7.747  | 21.425 | 14.586 | 9.671  | 93.537  |
| 65 | 3.105  | 10.164 | 6.634  | 4.991  | 24.913  |
| 66 | 4.505  | 10.285 | 7.428  | 2.891  | 8.357   |

|     |        |        |        |        |         |
|-----|--------|--------|--------|--------|---------|
| 67  | 1.356  | 8.916  | 4.575  | 3.903  | 15.230  |
| 68  | 4.505  | 10.285 | 7.428  | 2.891  | 8.357   |
| 69  | 9.946  | 16.290 | 13.118 | 4.486  | 20.127  |
| 70  | 0.803  | 20.101 | 9.775  | 9.720  | 94.479  |
| 71  | 17.999 | 23.130 | 20.564 | 3.628  | 13.164  |
| 72  | 4.505  | 10.285 | 7.428  | 2.891  | 8.357   |
| 73  | 2.227  | 6.798  | 3.761  | 2.629  | 6.914   |
| 74  | 0.693  | 11.045 | 5.974  | 5.179  | 26.822  |
| 75  | 0.760  | 6.537  | 3.088  | 3.047  | 9.286   |
| 76  | 0.212  | 16.687 | 8.775  | 8.257  | 68.183  |
| 77  | 17.649 | 22.013 | 20.288 | 2.322  | 5.390   |
| 78  | 3.776  | 3.977  | 3.876  | 0.142  | 0.020   |
| 79  | 5.833  | 29.374 | 14.650 | 12.834 | 164.711 |
| 80  | 8.976  | 12.508 | 10.753 | 1.766  | 3.119   |
| 81  | 17.715 | 21.730 | 20.283 | 2.230  | 4.974   |
| 82  | 2.227  | 6.798  | 3.761  | 2.629  | 6.914   |
| 83  | 4.934  | 14.821 | 9.138  | 5.107  | 26.077  |
| 84  | 0.640  | 14.973 | 6.567  | 7.482  | 55.975  |
| 85  | 0.466  | 1.505  | 0.985  | 0.734  | 0.539   |
| 86  | 6.402  | 18.647 | 11.078 | 6.615  | 43.757  |
| 87  | 1.269  | 10.855 | 5.701  | 4.834  | 23.363  |
| 88  | 2.349  | 14.748 | 8.548  | 8.768  | 76.873  |
| 89  | 0.808  | 9.649  | 5.228  | 6.251  | 39.079  |
| 90  | 4.272  | 17.037 | 8.844  | 7.112  | 50.577  |
| 91  | 4.422  | 20.536 | 10.915 | 8.501  | 72.263  |
| 92  | 3.370  | 27.527 | 15.448 | 17.081 | 291.770 |
| 93  | 4.505  | 10.285 | 7.428  | 2.891  | 8.357   |
| 94  | 2.227  | 6.798  | 3.761  | 2.629  | 6.914   |
| 95  | 3.846  | 14.296 | 8.065  | 5.508  | 30.337  |
| 96  | 0.466  | 1.505  | 0.985  | 0.734  | 0.539   |
| 97  | 8.329  | 24.474 | 15.306 | 8.293  | 68.768  |
| 98  | 3.918  | 8.674  | 6.296  | 3.363  | 11.310  |
| 99  | 2.227  | 6.798  | 3.761  | 2.629  | 6.914   |
| 100 | 3.573  | 22.588 | 10.272 | 10.680 | 114.063 |

[Back to top](#)

## Summary Report

| Entity           | MoE / MoP | # Replications | Min   | Max   | Mean  | StdDev | Variance |
|------------------|-----------|----------------|-------|-------|-------|--------|----------|
| SimpleStatsTally | 0         | 100            | 0.000 | 3.000 | 2.400 | 0.804  | 0.646    |

[Back to top](#)

---

## Conclusions and Recommendations

### *Conclusions*

The mean time spent by the arriving airplanes is less than the mean time spent by the departing airplanes

### *Recommendations for Future Work*

Investigation on the different distribution that take the plane to move from one location to the other could be a good subject for future work, in this simulation, the author varied the mean of the same type of distribution allocated to movement of the Airplanes

[Back to top](#)

---

This report was autogenerated by the Viskit Event Graph and Assembly modeling tool using Simkit discrete-event simulation (DES) libraries. Online at <https://diana.nps.edu/Viskit> and <https://diana.nps.edu/Simkit>.

THIS PAGE INTENTIONALLY LEFT BLANK

## LIST OF REFERENCES

- Akpan, Justice I., and Roger J. Brooks. 2005. Practitioners perception of the impacts of virtual reality on discrete-event simulation. Proceedings of the 2005 Winter Simulation Conference (December).
- Brutzman, D., and Daly. L. 2007. *X3D Extensible 3D Graphics for Web Authors*. Morgan Kaufmann.  
<http://x3dgraphics.com> (Accessed March 2008).
- Buss, A. H. 2004. *Simkit analysis workbench for rapid construction of modeling and simulation components*. Proceedings of the Fall Simulation Interoperability Workshop (September), Accessed 15 August 2006 at  
<http://www.sisostds.org/index.php?tg=fileman&idx=get&id=2&gr=Y&path=Simulation+InAteroperability+Workshops%2F2004+Fall+SIW%2F2004+Fall+SIW+Papers+and+Presentations&file=04F-SIW-020.pdf>.
- Buss, A. H., and P. J. Sanchez. 2005. *Simple Movement and Detection in Discrete Event Simulation*. Proceeding of the 2005 Winter Simulation Conference. Accessed 13 March 13, 2008 at <http://www.informs-cs.org/wsc05papers/118.pdf>.
- Buss, A. H., and R. Szechtman. 2006. *OA3302 System Simulation*, course notes, <http://diana.nps.edu/oa3302/Handouts/> Accessed March 2008.
- Law, A., and D. Kelton. 2000. *Simulation Modeling and Analysis*, Third Edition, McGraw Hill.
- Miller, J.A. *Simulation and Modeling in Java*. Accessed 17 January 2008 at <http://chief.cs.uga.edu/~jam/home/courses/csci4210/book.html>).
- Schruben, L. 1983. Simulation Modeling with Event Graphs. *Communications of the ACM* 26, : 957.
- Sullivan, P.J., *Evaluating the Effectiveness of Waterside Security Alternatives for Force Protection of Navy Ships and Installations Using X3D Graphics and Agent-Based Simulation*. Master's Thesis, Naval Postgraduate School, Monterey, California, September 2006.
- Wu, T. C., 2004. *An Introduction to Object-Oriented Programming with Java*. Third Edition. New York, New York: McGraw-Hill.
- \_\_\_\_\_. 2002. *Component Based Simulation Modeling With Simkit*. Proceeding of the 2002 Winter Simulation Conference.



- . 2001. *Basic Event Graph Modeling*. Simulation News Europe (April), Accessed 13 March 2008 at <http://diana.nps.edu/oa3302/Handouts/BasicEventGraphModeling.doc>.
- . 2000. *The Event List*. OA3302 System Simulation Winter 2007 class notes. Accessed 13 March 2008 at <http://diana.nps.edu/oa3302/Handouts/EventList.pdf>.

## INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center  
Ft. Belvoir, Virginia
2. Dudley Knox Library  
Naval Postgraduate School  
Monterey, California
3. Don Brutzman  
Naval Postgraduate School  
Monterey, California
4. Arnold Buss  
Naval Postgraduate School  
Monterey, California
5. Terry Norbraten  
Naval Postgraduate School  
Monterey, California
6. Byounghyun Yoo  
Naval Postgraduate School  
Monterey, California