DEVELOPMENT OF A VIRTUAL FACTORY ENVIRONMENT TO STUDY, SIMULATE AND IMPROVE THE MATERIAL FLOW BETWEEN MULTIPLE MICRO ASSEMBLY WORK CELLS

By

ABRAHAM ROBLEDO GALLEGOS

Bachelor of Science in Industrial Engineering

Universidad de las Américas

Puebla, Puebla

2007

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 2015

DEVELOPMENT OF A VIRTUAL FACTORY ENVIRONMENT TO STUDY, SIMULATE AND IMPROVE THE MATERIAL FLOW BETWEEN MULTIPLE MICRO ASSEMBLY WORK CELLS

Thesis Approved:

Dr. Terry Collins

Thesis Adviser

Dr. Camille DeYong

Dr. Jennifer Glenn

ACKNOWLEDGEMENTS

To God, who blesses me with access to knowledge that helps me to become a wiser man.

To my mother, Olivia Gallegos, who taught me, through her example, that love, faith and perseverance can make anything happen, no matter the circumstances.

To my father, Teodoro Robledo, whose words full of a working man's wisdom taught me that wise men are quiet and observe thoroughly before giving an intelligent opinion.

To my wife, Luana, my daughter Luiza and upcoming baby Gabriel, who inspire me with love every single day and who had patience during all those long nights and weekends that were dedicated to fulfill my academic objectives.

To my sister Olivia, who has been a loyal friend, companion, teammate, counselor and counselee since we used to play as little children in the guest bedroom upstairs.

To my company and professional mentors who have been fundamental in my development as an engineer, manager and leader. Rob LaRosa, Keith White, Robert Landi, Randy Walker, Rafael Colon, John Zeigle, Larry Golen and Victor Mora are among those who have made a difference in my professional life.

To Dr. José Pablo Nuño from UPAEP who inspired me to continue my education.

To Dr. Terry Collins, Dr. Camille DeYoung, Dr. Jeniffer Gleen, Dr. Sunderesh Heragu, Ms. Megan Rose Hughes and Mrs. Laura Lee Brown, from OSU, who supported me and gave me hope during the most difficult time in my academic life. They are an example of those who do the right thing regardless of what may come ahead.

Significant quotes that inspired me along the way:

"The greatest accomplishment is not in never falling, but in rising again after you fall" – Vince Lombardi

"The supreme quality for leadership is unquestionably integrity. Without it, no real success is possible, no matter whether it is on a section gang, a football field, in an army, or in an office" – Dwight D. Eisenhower

Name: ABRAHAM ROBLEDO GALLEGOS

Date of Degree: MAY 2015

Title of Study: DEVELOPMENT OF A VIRTUAL FACTORY ENVIRONMENT TO STUDY, SIMULATE AND IMPROVE THE MATERIAL FLOW BETWEEN MULTIPLE MICRO ASSEMBLY WORK CELLS

Major Field: INDUSTRIAL ENGINEERING & MANAGEMENT

Abstract: This thesis focuses on the creation of a virtual factory environment and its use to improve material flow between micro assembly work cells. The idea is to create a methodology that enables users to study the improvement of micro factory layouts and compare them based on their performance in the virtual environment. The main objectives for this project are to create a virtual factory environment using Unity® software, a game engine, and use it to simulate near optimal routing sequences between work cells.

Programming in Unity® creates user interfaces that accept inputs with near optimal sequences that visit all the micro assembly cells in the simulated factory. Near optimal sequences are obtained using genetic algorithms within the Global Optimization tool from MatLab. This tool calls pre - programed functions that repeatedly apply genetic operators, like crossovers and mutations, to a given sequence in order to find a near optimal one. MatLab feeds from external data that consists of the distances between the work cells. These distances are calculated and stored in an Excel file which is read directly from the MatLab environment.

All thesis objectives are fulfilled and the proposed methodology is used successfully to create a virtual micro factory in Unity®. The model is used to simulate several sequences for different circumstances:

- Material distribution for a twenty four cell layout connected by conveyors.
- Material distribution for only twelve of the twenty four available stations.
- Design of material distribution sequence to supply twenty four work cells that are not limited by conveyor connections.

In all situations the cumulative travel distances calculated in the Unity® model matched the objective function value estimated in MatLab. This validated the ability of the model to accurately represent the motion of materials within a micro assembly factory.

This methodology can be used not only to study and improve existing micro factory systems but to also design future micro factories to be more efficient. The flexibility of the Unity® environment enables the users not to only simulate the movement of materials along near optimal sequences but to also reposition objects to quickly create different layout options.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

CHAPTER I

INTRODUCTION

1.1 Introduction to Micro-assembly and virtual prototyping.

Micro assembly is an emerging field that refers to assembling structures with micron-sized components. These small components are manipulated by micro grippers and micro position manipulators that are integrated to a micro assembly work cell. Virtual Reality technology has been successfully used to create prototypes of micro assembly work cells [1]. These prototypes enable researchers and users to better understand the characteristics of the micro devices that are assembled with the work cells. The prototypes also help to better comprehend the structure and functions of the micro assembly cell. The sequence of micro assembly steps, the material handling moves, and the anti-collision path planning are some examples of concepts that can be better seen and studied using a virtual prototype.

1.2 Problem statement and motivation

Although a lot of work and research has been conducted at a micro work cell level; there are still needs and opportunities to study micro assembly processes at a factory level. In a factory setting, several cells work sequentially to assemble a micro product; the material flows from one cell to the other with the assistance of a conveyance system.

If a virtual factory prototype with multiple micro assembly work cells is created, the relationship between the work cells, the flow of parts between them, and the overall factory performance can be more easily explored and improved. With such virtual factory environment, the following can be studied and/or improved:

- Time and distance that is consumed by the routes that parts use to flow between cells.
- Micro assembly factory balancing including both, work cell processing times and material flow times.
- Material path planning to avoid collisions.
- Use of genetic algorithms and other methods to improve the micro assembly process.

This thesis focuses on the creation of the virtual factory environment and the use of genetic algorithms and other methods to improve material flow between micro assembly work cells. The idea is to create a methodology that allows for improvement and comparison of micro factory layouts based on efficiency and performance.

1.3 Goals and objectives

The main objective for this project is to create a virtual factory environment and use it to study and improve micro assembly processes. The following objectives derive from the main one:

- Create a virtual factory environment using computer graphics (UNITY® software).
- Explore the potential use of genetic algorithms and other methods to improve the routing between work cells.

1.4 Overview of proposed approach

This project uses virtual reality and genetic algorithms to study and improve a micro assembly factory. In this section we explain the concepts of virtual reality and genetic algorithms and how they are applied to complete the objectives listed in previous sections.

Virtual reality relies on computers to create virtual prototypes or models in simulated scenarios that enable humans to study, analyze, interact, and experiment with them in order to infer possible behaviors of real-world systems. Computer-aided design (CAD) and simulation software are good examples of the technology that companies use nowadays to have more efficient design processes. Virtual reality technology can be used along with CAD and simulation principles to even immerse the users in a computer simulated environment. This enables users to detect and solve issues earlier in design stages. Similarly, this technology allows for better studying and understanding of structures that are difficult to be perceived by human senses. For instance, Virtual Reality is used to create virtual prototypes of Micro Assembly environments and devices that would never be easily seen by the human eye. The simulated environment enables users to see and interact with micron sized parts and equipment.

Genetic algorithms use an initial randomly generated arrangement of characters and transforms it through the application of genetic operators. The application of such operators may help the initial arrangement to evolve towards something better. Among the most commonly known operators we find the crossover, the mutation and the inversion. When the initial arrangement of characters represents an assembly or routing sequence, the genetic algorithms can be used to continuously search for a better sequence or route. In a Micro assembly factory a better route would be the one that reduces the time and/or amount of distance traveled by the material between work cells.

1.5 Remarks

Studies in the Micro Assembly world have mostly focused at the work cell level; however, there have not been a lot of studies done at the factory level. This study proposes to use virtual reality and genetic algorithms to create a virtual factory prototype where the flow of materials between cells can be studied and improved by applying genetic operators to obtain material distribution sequences that enhance micro assembly layout performance.

CHAPTER II

LITERATURE REVIEW

2.1. Introduction

This chapter is a literature review of work that has been done in micro assembly and virtual reality domains. Information related to micro assembly papers is presented first followed by information related to virtual reality papers. Finally, a summary of papers related to the application of virtual reality in the field of micro assembly is provided.

2.2. Literature related to micro assembly

In [2], Shet et. al. review several micro assembly techniques used for the integration of microelectromecanical structures (MEMS) including selective area growth, flip-chip bonding, epitaxial lift-off, electrostatic alignment, fluidic self-assembly (FSA), and magnetically assisted statistical assembly (MASA); however, they also recognize the fact that these techniques have some limitations and that there is still need for further development in this field. The authors propose the magnetic field assisted assembly process (MFAA) which prepares substrates and micro components separately and uses a magnetic field to assist with the integration of the micro components.

In [3], Cecil et. al. provide a review micro gripping and manipulation techniques, micro assembly systems, and computer and information technology –based approaches including the use of virtual reality for micro assembly. The authors recognize that MEMS technology involves little assembly tasks and see the need to develop techniques to efficiently and rapidly assemble micro devices that are composed of different materials or complex geometries. They define Micro-Assembly as the field that focuses on the assembly of tiny, micron-sized parts that need to be handled in order to be attached to the final micro product. The authors summarize three current challenges in the field of micro assembly: the design of high-speed assembly techniques and mechanisms, the development of high-fidelity virtual reality-based simulation environments, and the need to develop technology that enable users in distributed systems to support micro assembly technology from geographically dispersed locations.

In [4], Ohba et. al. an observational sensor system for tele-micro-operation is proposed. The authors use this smart sensor system and the concept of 'depth in focus' to simultaneously reconstruct 3D virtual reality micro environments with all-in-focus images captured by dynamic focusing lens and smart camera systems.

In [5], Beltrami et. al. use flexures and joints to create a multi degree of freedom micro machine. The authors state that flexures have advantages that are helpful for the design of precision robots. They explain that a flexure joint links two solids permitting motion in one direction but not in others; which enables them to perform high precision tasks.

In [6], Qin discusses micro forming operations and miniature manufacturing systems. The author summarizes current developments in the field around the world and recognized the need to reduce the scale of the equipment used to manufacture miniature products. The author summarizes scientific, technological, social, and economical issues associated to the development of micro-factories. One of the conclusions is that while the micro factories and machines developed to-date

have been focused on the demonstration of capabilities, further developments should consider the practicability of the machines/systems and take the industrial application into account.

In [7], Wang et. al. issues with the visual controllers used in visual serving are discussed. The authors propose a proportional differential (PD) visual controller to improve the dynamic performance of visual servoing in micro assembly. Genetic algorithms are used to obtain near optimal controller parameters. The concept is integrated to a micro assembly cell that inserts a micro peg into a hole using micro grippers.

In [8], Feddema et. al. discuss the roll of van der Waals forces in micro assembly planning. The authors report that with micron-sized parts the effect of interactive forces, such as van de Waals, electrostatic and adhesive forces, is significant. Because of this phenomenon, micro assembly plans are not reversible; meaning that motions required to pick up a part are not the reverse motions to release a part. In the micro world gravity is negligible and interactive forces dominate. Authors recommend that the influence of these forces is considered when planning micro assembly tasks. Authors also recognize the need for further research that helps computing these forces between different micro component shapes.

In [9], Hollis et. al. propose a platform technology for micro assembly and name it Agile Assembly Architecture (AAA). The authors define platform technologies as those that serve as springboards for other technologies and are essential for progress in multiple fields. Agile Assembly Architecture has two components: a distributed system of tightly integrated mechanical/computational agents which consist of a collection of modular robots, feeders and fastening/assembly equipment, and an unified Interface Tool that allows a user to select among the available agents, to order them in the assembly system and to operate them in both, a virtual environment and in the real world. The authors visualize a highly modular and scalable robotic pipeline approach to design micro assembly systems that are able to assembly micro products of

different sizes, to integrate diverse fastening technologies, to use parallel assembly stations, and to assembly complex products that require a large number of process steps.

In [10], Hollies et. al. present a mini-factory that was created the Microdynamic Systems Laboratory at Carnegie Mellon University applying the Agile Assembly Architecture discussed in the previous paragraph. The first mini-factory successfully integrated various precision manufacturing processes to assemble capacitors at a rate of one finished product every twenty-five seconds. A second mini-factory was improved to increase throughput by adding parallel processes. The rate of production after the improvements completed one capacitor every nine seconds.

In [11], Gaugel et. al. present the MiniProd system which uses compact process and assembly modules that are mounted on a platform using standardized interfaces. The core element of each micro assembly segment is a planar motor that has been designed for miniaturized applications and is responsible for feeding the parts and controlling the flow of materials between individual stations. The authors propose the use of a virtual module kit system to operate and configure the system. When users use this kit, they are able to select the required assembly modules and to integrate them in the virtual environment to form a mini-factory. The authors state that by simulating the system, the amount of time to bring it into operation is significantly reduced.

Some conveyance systems for micro assembly have also been proposed. For example, in [12], Ferreira proposes planar piezoelectric micro conveyers to move micro objects relying on the cooperation of arrayed direct-drive standing wave ultrasonic motors. Also in [13], Li et. al. propose an omni-directional mobile micro robot for material handling within a micro factory. The micro robot is capable to move linearly and to steer in order to position its micro gripper in the required positions.

Efforts to improve micro assembly system performance have been reported. For example, in [14], Subramaniyam et. al. use a methodology to improve efficiency of micro assembly systems by analyzing alternative layouts. Also, in [15], Shuang et. al. discuss the micro assembly sequence planning problem and propose the use of a modified particle swarm/ant colony optimization algorithm (hybrid PS-ACO algorithm) to solve it. Authors compare the optimization results with those obtained with genetic algorithms (GA) and the regular ant colony optimization algorithm (ACO).

2.3. Literature related to virtual reality

In [16], Cecil et. al. define a virtual prototype (VP) as a three-dimensional virtual reality based model that mimics a target object, system or environment. The authors define virtual prototyping as the process to create and use a VP. The authors define the characteristics of a virtual prototype: a VP must possess accurate geometry, topology, and appearance; should be capable of simulate object or environment characteristics, including real-time responses; is a computer based representation; and must possess the ability to interface with VR technology and support immersive applications. The authors present a summary of different applications of virtual prototyping for product and process design. They also suggest that the use of VPs facilitates concurrent engineering and the earlier identification of problems and solutions in product and process design projects.

In [17], Jayaram et. al. propose and implement a Virtual Assembly Design Environment (VADE) prototype to enable engineers to detect assembly issues earlier in the design stage. Authors state that the use of VR can benefit companies by reducing product development and fabrication time, introducing technology and design methods faster, improving product quality and reliability, and reducing cost.

In [18], Gomes de Sá et. al. explain the application of virtual reality technology in the automotive industry to assist product development and present examples of projects implemented for verification of assembly and maintenance processes.

In [19], Wang et. al. propose a dynamic data model for visualization of industrial assemblies. Their model integrates spatial and dynamic data to accurately represent the changes and movements of assembly structures. In this model, the assembly structure does not contain a real geometry; it only contains references to part geometries. This reference information is what defines the assembly structures. The authors state that a virtual assembly system not only allows for visualization and manipulation of assembly structures, but also enables simulation, verification, evaluation of assembly processes, construction of virtual prototypes and training.

In [20], Weidlich et. al. summarize several examples of virtual reality approaches for immersive design. The authors state that researchers have implemented three prototypical ways of immersive design: linking a VR system and a CAD core, linking a VR system and a CAD system, and using voxel models for geometry description. Examples of each of these methods are presented in the paper.

In [21], Lin et. al. propose a virtual factory that provides an efficient prototyping test bed. The authors model their virtual factory with multiple processes and libraries that work together. Their approach considers a physical domain that refers to the real assembly cells, and a pseudo domain that refers to models of the real cells. The system is capable of virtual automation and it is provided with libraries and specification tools.

In [22], Wenbin et. al. propose a production engineering-oriented virtual factory (PEOVF) to assist with the design of processes and reduce the amount of changes that are necessary due to the lack of systemic and concurrent approaches when designing processes.

In [23], Renard et. al. present a really innovative approach to interact with virtual environments through the use of brain-computer interfaces (BCIs). BCIs are communication systems that enable users to interact with computers solely by brain activity. In this research, users manipulate virtual environments by imagining hand or foot motions.

2.4. Literature related to the application of virtual reality in the field of micro assembly

In [24], Gobinath et. al. propose an Integrated Physical and Virtual Reality (VR) environment to aid in the assembly of micro devices. The approach considers three functional areas: creation of simulated environments using virtual reality (VR) technology, development of a VR-linked physical environment to support the rapid assembly of micro devices, and the development of an information oriented model that enables the accomplishment of the other two functional areas. The proposed virtual micro assembly work cell consists of a user interface module (UIM), a visualization manager (VM), a part initialization module (PIM), a navigation module, an assembly manager, an immersive part handling module, and a collision detection module. The authors also apply generic algorithms to near optimal generation of micro assembly cell sequences.

In [25], Ferreira et. al. propose an automated micro manipulation work cell that is operated with the assistance of vision servoing and virtual reality. The authors present a pushing-based micromanipulation strategy and a micro handling strategy which are supported by vision feedback and virtual environments. The proposed cell positions, handles and assembles the components thanks to a collection of micromanipulators and grippers that interact during the assembly process.

In [26], Cecil et. al. further discuss the development of a virtual and physical work cell to assemble micro devices and explain in detail the application of genetic algorithms to generate

near optimal assembly sequences. The algorithm generates initial sequences randomly, checks for uniqueness in each sequence, performs hybrid crossover operations, checks for precedence constraints, performs mutation operations for sequences, checks for precedence constraints again, selects new sequences and repeats the process until a near optimal solution is identified.

In [27], Ferreira et. al. propose a micro assembly desktop station that is supervised by virtual reality to perform micro positioning, handling and assembly tasks. The micro factory concept is integrated by two motion micro manipulators equipped with a micro tool holder, and a coarse motion worktable. This constitutes a multi degree of freedom fine positioning system. The system is overseen by an Optical Microscope. The proposed system enables the operator to determine the tasks to be performed by using a virtual reality guiding-system interface. The detailed scenario is built with CAD-CAM databases which enables the system to perform the tasks while avoiding collisions.

Finally, in [28], Cecil et. al. discuss the need to develop an advanced virtual reality environment to support the study of factory and work cell level assembly alternatives. The design of a Virtual Factory Environment (VFE) should be able to compare factory level alternatives and determine feasible ways to assemble a given design. The authors implemented a VFE using Coin 3D and C++ tools. The factory analysis module is capable of generating assembly sequence and path plans. This VFE can be used to enable comparison of factory and work cell assembly alternatives.

CHAPTER III

PROPOSED METHODOLOGY

3.1 Introduction

This project uses Unity software, a game engine, to create a virtual environment of a micro assembly factory with multiple cells. This environment simulates the motion of raw material from cell to cell. The User Interface is integrated in Unity and only requires inputting a sequence of steps to simulate a path that delivers material to each one of the assembly cells. Excel and MatLab are used along with Unity to make this simulation possible.

3.2 Interaction of Unity, Excel and MatLab

Using programming in Unity allows us to create a user interface that accepts a near optimal sequence to visit all the micro assembly cells in the simulated factory. This sequence is obtained from the results of a MatLab exercise. MatLab has a Global Optimization tool that can be used with Genetic Algorithms to solve problems. The Global Optimization tool works by calling several pre - programed functions that apply genetic operators, such as crossovers and mutations to a given initial sequence. The outcome of this exercise is a near optimal sequence that can be simulated in the virtual environment.

The MatLab algorithm solver tool feeds from external data before being able to find a near optimal sequence. This external data consists of the distances between coordinates of the work cell platforms in the unity environment. These distances can be calculated and stored in an Excel file which is read directly from the MatLab environment.

3.3 Unity and Virtual Reality Modeling

Environments and objects in Unity can be built with resources provided by the game engine or can be exported from files created with CAD software that are converted to a '.3ds' extension. Once the micro assembly cell models are created or imported into Unity, the simulation part is accomplished with scripting. The behavior of objects and responses to user inputs are possible thanks to the scripts that are attached to objects within the virtual environment [29].

"Unity supports three programing languages natively:

- C# (pronounced C-sharp), an industry-standard language similar to Java or C++;
- UnityScript, a language designed specifically for use with Unity and modelled after JavaScript."[29]

This project uses UnityScript to create an interphase that enables users to input near optimal material distribution sequences and to simulate the material movement throughout the micro factory.

This project uses UnityScript to enable users to input near optimal material distribution sequences, to control the movement and positioning of the objects, and to simulate the near optimal material traveling path between micro assembly cells.

3.4 User Interface

In Unity, scripts are attached to objects. The objects are listed in the Hierarchy window. If an object is selected or highlighted in this window, the associated scripts and user required inputs are listed in the Inspector window under the script section [29]. See Figure 1.

To input sequences the user needs to highlight the object that represents the pin pallet in the Hierarchy window which is named "Sphere Pallet." Once the object is selected, the Inspector window displays the fields that store the order of coordinates that will be visited by the pallet. The users only have to drag the objects that represent each micro-assembly cell into each one of the sequence fields and the scripting will visit those cells one by one in the order that the user dragged them into the fields. See Figures 2 and 3 below. More details on the specific techniques used to build the model in Unity will be given in Chapter IV.

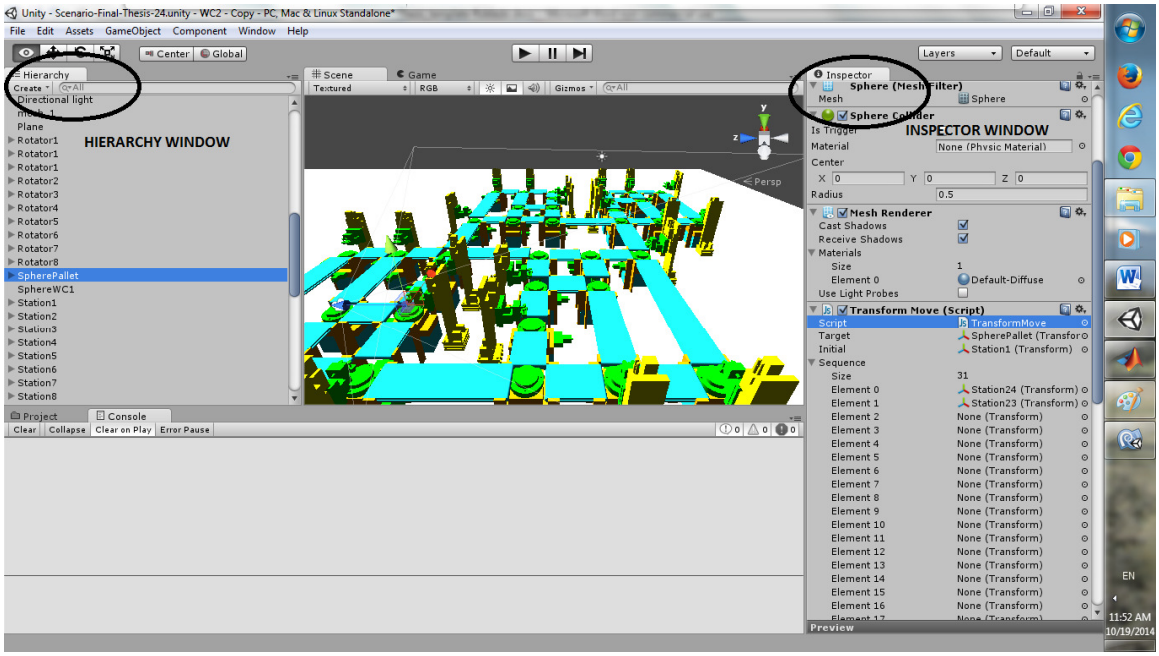

Figure 1. Hierarchy and Inspector windows

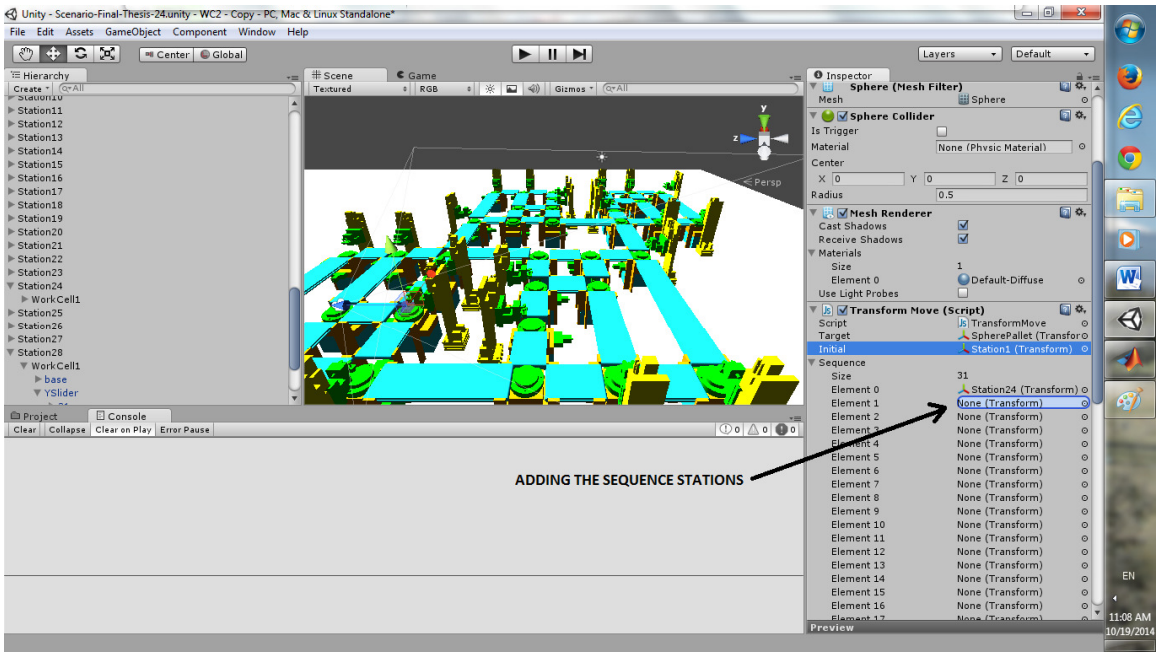

Figure 2. Adding Sequence Stations to Pin Pallet object

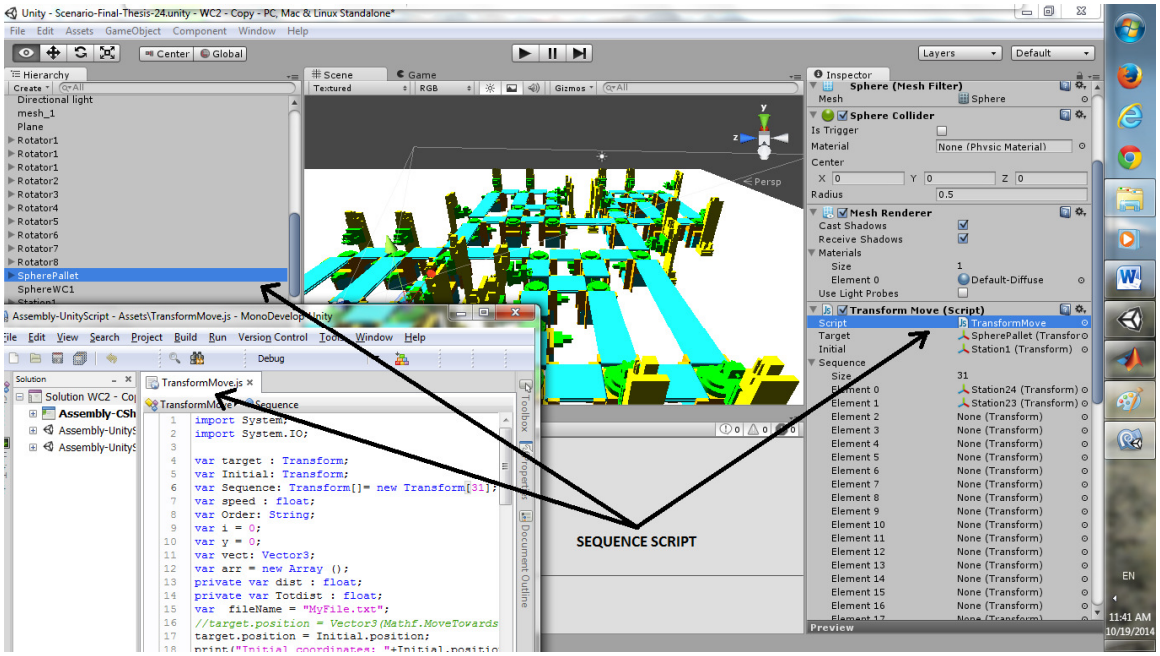

Figure 3. Sequence script is attached to Pin Pallet object

3.5 Excel and MatLab coordinates input

The sequence that is input into Unity is obtained from a genetic algorithm solver exercise performed by the Global Optimization Tool in MatLab [30, 31]. The data needed to run the solver is extracted from Excel where a file stores all the distances between micro-assembly cells coordinates. This file processes the different cell positions to create a distance matrix that can be read by MatLab scripting when running its solver.

Two Excel files have been created to support the process in MatLab. The first file calculates all the distances based on cell coordinates and the second file is only to import the distance Matrix into MatLab.

The "Positions" tab and the "Rotators-Positions" tab in the first Excel file contain the coordinates where material is delivered to each micro-assembly cell and the coordinates of the rotators where the material pallet could change direction in order to continue its path. See Table 1 below.

Table 1. Position to Position distance matrix

POSITION TO POSITION				Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
				X	5.71	5.71	-6.71	-6.71	-6.71	-1.83	2.20	14.05	26.51	26.51	23.00	18.97	17.61	21.63	26.51	26.51	26.51	14.09	14.09	5.71	-6.71	-6.71	-3.20	0.83
				Y	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55
Position	X	Y	Z	Z	-13.35	-8.97	-8.99	-13.35	-17.80	-17.80	-17.80	-17.80	-17.80	-22.18	-26.63	-26.63	-34.36	-34.36	-34.36	-38.81	-43.19	-43.19	-38.81	-34.36	-34.36	-29.98	-25.53	-25.53
					1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	5.71	-2.55	-13.35	1	0.00	4.39	13.17	12.42	13.19	8.75	5.67	9.44	21.27	22.60	21.79	18.76	24.14	26.36	29.56	32.87	36.37	31.00	26.80	21.01	24.41	20.76	15.09	13.12
2	5.71	-2.55	-8.97	2	4.39	0.00	12.42	13.17	15.24	11.61	9.50	12.14	22.60	24.64	24.71	22.08	28.04	29.97	32.83	36.37	40.05	35.24	31.00	25.40	28.28	24.41	18.81	17.27
3	-6.71	-2.55	-8.99	3	13.17	12.42	0.00	4.37	8.81	10.07	12.53	22.55	34.37	35.75	34.55	31.16	35.14	38.04	41.80	44.64	47.68	40.03	36.36	28.25	25.38	20.99	16.92	18.18
4	-6.71	-2.55	-13.35	4	12.42	13.17	4.37	0.00	4.44	6.60	9.95	21.22	33.52	34.37	32.54	28.91	32.13	35.28	39.30	41.85	44.65	36.37	32.87	24.41	21.01	16.63	12.68	14.33
5	-6.71	-2.55	-17.80	5	13.19	15.24	8.81	4.44	0.00	4.88	8.91	20.75	33.22	33.51	30.99	27.16	29.42	32.82	37.12	39.30	41.81	32.83	29.56	20.70	16.57	12.18	8.50	10.80
6	-1.83	-2.55	-17.80	6	8.75	11.61	10.07	6.60	4.88	0.00	4.03	15.87	28.34	28.68	26.35	22.60	25.54	28.72	32.82	35.28	38.05	29.97	26.36	18.20	17.27	13.12	7.86	8.18
7	2.20	-2.55	-17.80	7	5.67	9.50	12.53	9.95	8.91	4.03	0.00	11.85	24.32	24.71	22.60	18.96	22.62	25.54	29.42	32.13	35.16	28.04	24.14	16.94	18.81	15.09	9.43	7.86
8	14.05	-2.55	-17.80	8	9.44	12.14	22.55	21.22	20.75	15.87	11.85	0.00	12.47	13.22	12.57	10.11	16.95	18.22	20.73	24.43	28.29	25.40	21.01	18.55	26.56	24.07	18.90	15.32
9	26.51	-2.55	-17.80	9	21.27	22.60	34.37	33.52	33.22	28.34	24.32	12.47	0.00	4.39	9.50	11.61	18.81	17.27	16.57	21.01	25.40	28.27	24.41	26.60	37.13	35.39	30.70	26.82
10	26.51	-2.55	-22.18	10	22.60	24.64	35.75	34.37	33.51	28.68	24.71	13.22	4.39	0.00	5.67	8.75	15.09	13.12	12.18	16.63	21.01	24.41	20.76	24.11	35.39	34.13	29.90	25.90
11	23.00	-2.55	-26.63	11	21.79	24.71	34.55	32.54	30.99	26.35	22.60	12.57	9.50	5.67	0.00	4.03	9.43	7.86	8.50	12.68	16.94	18.81	15.09	18.94	30.70	29.90	26.22	22.20
12	18.97	-2.55	-26.63	12	18.76	22.08	31.16	28.91	27.16	22.60	18.96	10.11	11.61	8.75	4.03	0.00	7.86	8.18	10.80	14.32	18.20	17.27	13.12	15.36	26.82	25.90	22.20	18.18
13	17.61	-2.55	-34.36	13	24.14	28.04	35.14	32.13	29.42	25.54	22.62	16.95	18.81	15.09	9.43	7.86	0.00	4.03	8.91	9.95	12.54	9.50	5.67	11.90	24.32	24.71	22.60	18.96
14	21.63	-2.55	-34.36	14	26.36	29.97	38.04	35.28	32.82	28.72	25.54	18.22	17.27	13.12	7.86	8.18	4.03	0.00	4.88	6.60	10.09	11.61	8.75	15.92	28.34	28.68	26.35	22.60
15	26.51	-2.55	-34.36	15	29.56	32.83	41.80	39.30	37.12	32.82	29.42	20.73	16.57	12.18	8.50	10.80	8.91	4.88	0.00	4.44	8.83	15.24	13.19	20.80	33.22	33.51	30.99	27.16
16	26.51	-2.55	-38.81	16	32.87	36.37	44.64	41.85	39.30	35.28	32.13	24.43	21.01	16.63	12.68	14.32	9.95	6.60	4.44	0.00	4.39	13.17	12.42	21.27	33.52	34.37	32.54	28.91
17	26.51	-2.55	-43.19	17	36.37	40.05	47.68	44.65	41.81	38.05	35.16	28.29	25.40	21.01	16.94	18.20	12.54	10.09	8.83	4.39	0.00	12.42	13.17	22.60	34.37	35.75	34.56	31.17
18	14.09	-2.55	-43.19	18	31.00	35.24	40.03	36.37	32.83	29.97	28.04	25.40	28.27	24.41	18.81	17.27	9.50	11.61	15.24	13.17	12.42	0.00	4.39	12.17	22.60	24.64	24.71	22.08
19	14.09	-2.55	-38.81	19	26.80	31.00	36.36	32.87	29.56	26.36	24.14	21.01	24.41	20.76	15.09	13.12	5.67	8.75	13.19	12.42	13.17	4.39	0.00	9.48	21.27	22.60	21.79	18.76
20	5.71	-2.55	-34.36	20	21.01	25.40	28.25	24.41	20.70	18.20	16.94	18.55	26.60	24.11	18.94	15.36	11.90	15.92	20.80	21.27	22.60	12.17	9.48	0.00	12.42	13.17	12.54	10.09
21	-6.71	-2.55	-34.36	21	24.41	28.28	25.38	21.01	16.57	17.27	18.81	26.56	37.13	35.39	30.70	26.82	24.32	28.34	33.22	33.52	34.37	22.60	21.27	12.42	0.00	4.39	9.50	11.61
22	-6.71	-2.55	-29.98	22	20.76	24.41	20.99	16.63	12.18	13.12	15.09	24.07	35.39	34.13	29.90	25.90	24.71	28.68	33.51	34.37	35.75	24.64	22.60	13.17	4.39	0.00	5.67	8.75
23	-3.20	-2.55	-25.53	23	15.09	18.81	16.92	12.68	8.50	7.86	9.43	18.90	30.70	29.90	26.22	22.20	22.60	26.35	30.99	32.54	34.56	24.71	21.79	12.54	9.50	5.67	0.00	4.03
24	0.83	-2.55	-25.53	24	13.12	17.27	18.18	14.33	10.80	8.18	7.86	15.32	26.82	25.90	22.20	18.18	18.96	22.60	27.16	28.91	31.17	22.08	18.76	10.09	11.61	8.75	4.03	0.00

Distances between cells are calculated in Excel using the Euclidean distance, Pythagoras Theorem, or commonly known coordinates distance formula [32-34]. For those cells connected by a straight path, the distance between them is simply the result of applying the distance formula with their coordinates. On the other hand, for those cells that are connected with paths that involve corners or rotators, the calculation is broken down in sections so that the straight segment distances are added regardless of the turns or changes of direction. For example, in the proposed layout, the path between cell 1 and cell 7 includes a 90 degree turn; the distance between them is calculated by adding the distances between cell 1 and the 90 degree turn and between cell 7 and the 90 degree turn.

The formula to calculate distances in Excel follows [34]:

=SQRT((($B6-F$2)^2)+(($C6-F$3)^2)+(($D6-F$4)^2))

The same formula is expressed for general use with two points (x_i,y_i,z_i) as [35]:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$.

Similarly to the "Positions" tab, the "Rotators-Positions" tab calculates the distances between the work cells and rotators that are connected through a straight path. Furthermore, there is a "Rotators-Rotators" tab that calculates the distances between the rotators that are connected through a straight path. See Tables 2 and 3 below.

Table 2. Position to Rotator Distance Matrix

POSITION TO ROTATOR				Rotator	1	2	3	4	5	6	7	8	9	10	11	12
				X	5.714	-1.827	26.514	18.974	14.090	21.631	-6.711	0.830	14.094	14.094	5.710	5.710
				Y	-2.552	-2.552	-2.552	-2.552	-2.552	-2.552	-2.552	-2.552	-2.552	-2.552	-2.552	-2.552
Position	X	Y	Z	Z	-17.797	-13.352	-26.626	-22.181	-34.365	-38.808	-25.534	-29.979	-22.181	-26.626	-25.534	-29.979
					1	2	3	4	5	6	7	8	9	10	11	12
1	5.714	-2.552	-13.352	1	4.44	7.54	24.68	15.93	22.62	30.02	17.40	17.33	12.17	15.70	12.18	16.63
2	5.714	-2.552	-8.966	2	8.83	8.72	27.29	18.72	26.74	33.82	20.71	21.57	15.65	19.55	16.57	21.01
3	-6.707	-2.552	-8.986	3	15.23	6.55	37.61	28.87	32.81	41.14	16.55	22.31	24.63	27.27	20.69	24.39
4	-6.707	-2.552	-13.352	4	13.19	4.88	35.77	27.16	29.56	38.09	12.18	18.26	22.60	24.68	17.39	20.75
5	-6.707	-2.552	-17.797	5	12.42	6.60	34.37	26.05	26.59	35.28	7.74	14.33	21.26	22.60	14.63	17.39
6	-1.827	-2.552	-17.797	6	7.54	4.44	29.68	21.26	22.98	31.49	9.15	12.47	16.51	18.21	10.80	14.33
7	2.199	-2.552	-17.797	7	3.52	6.00	25.87	17.34	20.39	28.62	11.80	12.26	12.68	14.81	8.50	12.68
8	14.048	-2.552	-17.795	8	8.33	16.48	15.28	6.60	16.57	22.34	22.15	17.98	4.39	8.83	11.38	14.76
9	26.514	-2.552	-17.795	9	20.80	28.69	8.83	8.72	20.71	21.57	34.11	28.43	13.17	15.24	22.20	24.11
10	26.514	-2.552	-22.181	10	21.26	29.68	4.45	7.54	17.40	17.33	33.39	26.84	12.42	13.19	21.07	22.22
11	22.999	-2.552	-26.626	11	19.41	28.15	3.52	6.00	11.80	12.26	29.73	22.42	9.95	8.91	17.32	17.61
12	18.974	-2.552	-26.626	12	15.93	24.68	7.54	4.44	9.15	12.47	25.71	18.45	6.60	4.88	13.31	13.68
13	17.605	-2.552	-34.363	13	20.39	28.62	11.80	12.26	3.52	6.00	25.87	17.34	12.68	8.50	14.81	12.68
14	21.631	-2.552	-34.363	14	22.97	31.49	9.15	12.47	7.54	4.44	29.68	21.26	14.32	10.80	18.20	16.51
15	26.510	-2.552	-34.363	15	26.59	35.28	7.74	14.32	12.42	6.60	34.37	26.05	17.39	14.63	22.60	21.26
16	26.510	-2.552	-38.808	16	29.56	38.09	12.18	18.25	13.19	4.88	35.77	27.16	20.75	17.39	24.67	22.60
17	26.510	-2.552	-43.193	17	32.83	41.15	16.57	22.32	15.24	6.56	37.62	28.88	24.41	20.70	27.29	24.64
18	14.090	-2.552	-43.193	18	26.74	33.82	20.71	21.57	8.83	8.72	27.29	18.72	21.01	16.57	19.55	15.65
19	14.090	-2.552	-38.808	19	22.62	30.02	17.40	17.33	4.44	7.54	24.67	15.93	16.63	12.18	15.70	12.17
20	5.710	-2.552	-34.365	20	16.57	22.32	22.20	18.01	8.38	16.53	15.24	6.56	14.79	11.41	8.83	4.39
21	-6.711	-2.552	-34.365	21	20.71	21.57	34.11	28.43	20.80	28.69	8.83	8.72	24.11	22.20	15.24	13.17
22	-6.711	-2.552	-29.979	22	17.40	17.33	33.39	26.84	21.26	29.68	4.44	7.54	22.22	21.07	13.19	12.42
23	-3.195	-2.552	-25.534	23	11.80	12.26	29.73	22.42	19.41	28.15	3.52	6.00	17.61	17.32	8.91	9.95
24	0.830	-2.552	-25.534	24	9.15	12.47	25.71	18.45	15.93	24.67	7.54	4.44	13.68	13.31	4.88	6.60

Table 3. Rotator to Rotator Distance Matrix

ROTATOR TO ROTATOR				Rotator	1	2	3	4	5	6	7	8	9	10	11	12
				X	5.713849	-1.82688	26.51447	18.97371	14.08998	21.63072	-6.7106	0.830146	14.09398	14.09398	5.709882	5.709882
				Y	-2.55176	-2.55176	-2.55176	-2.55176	-2.55176	-2.55176	-2.55176	-2.55176	-2.55176	-2.55176	-2.55176	-2.55176
Rotator	X	Y	Z	Z	-17.7965	-13.3516	-26.626	-22.181	-34.3647	-38.8075	-25.5339	-29.9789	-22.181	-26.626	-25.5339	-29.9789
					1	2	3	4	5	6	7	8	9	10	11	12
1	5.713849	-2.55176	-17.7965	1	0	8.753304	22.59702	13.96594	18.56515	26.35922	14.63673	13.12478	9.457801	12.17315	7.737381	12.18237
2	-1.82688	-2.55176	-13.3516	2	8.753304	0	31.29605	22.59698	26.36095	34.61596	13.12479	16.83827	18.20529	20.72881	14.32524	18.25572
3	26.51447	-2.55176	-26.626	3	22.59702	31.29605	0	8.753337	14.63749	13.12405	33.24302	25.90225	13.19192	12.42049	20.83323	21.07304
4	18.97371	-2.55176	-22.181	4	13.96594	22.59698	8.753337	0	13.12608	16.83747	25.90223	19.7483	4.87973	6.600715	13.68105	15.38623
5	14.08998	-2.55176	-34.3647	5	18.56515	26.36095	14.63749	13.12608	0	8.752208	22.59752	13.96635	12.18375	7.738761	12.17413	9.458411
6	21.63072	-2.55176	-38.8075	6	26.35922	34.61596	13.12405	16.83747	8.752208	0	31.29568	22.59666	18.25498	14.32455	20.72829	18.20488
7	-6.7106	-2.55176	-25.5339	7	14.63673	13.12479	33.24302	25.90223	22.59752	31.29568	0	8.753321	21.07303	20.83322	12.42048	13.19191
8	0.830146	-2.55176	-29.9789	8	13.12478	16.83827	25.90225	19.7483	13.96635	22.59666	8.753321	0	15.38624	13.68105	6.600719	4.879736
9	14.09398	-2.55176	-22.181	9	9.457801	18.20529	13.19192	4.87973	12.18375	18.25498	21.07303	15.38624	0	4.44499	9.029687	11.44992
10	14.09398	-2.55176	-26.626	10	12.17315	20.72881	12.42049	6.600715	7.738761	14.32455	20.83322	13.68105	4.44499	0	8.454921	9.029687
11	5.709882	-2.55176	-25.5339	11	7.737381	14.32524	20.83323	13.68105	12.17413	20.72829	12.42048	6.600719	9.029687	8.454921	0	4.44499
12	5.709882	-2.55176	-29.9789	12	12.18237	18.25572	21.07304	15.38623	9.458411	18.20488	13.19191	4.879736	11.44992	9.029687	4.44499	0

In order to accurately calculate all the distances of a given work cell layout the Excel file uses different tabs that are interconnected by formulas to come up with the final distances matrix. The tabs use a binary input method for the user to define which work cells directly connect with others with a straight path, which ones connect with others through rotators, and which ones do not connect at all without going through others. In Tables 4, 5 and 6 below it can be seen how a one (1) is input by the user when there is a straight path between two cells, between a cell and a rotator or between two rotators. These inputs (ones) are later multiplied by the values on the

distance matrices to assign an actual distance value to the matrix that is later exported to MatLab. If a one is input by the user, the final matrix will have a value coming from the distances matrices. If a zero is input by the user, no value will be added to the final matrix for MatLab.

Table 4. Binary Position-to-Position Matrix for user input

POSITION TO POSITION (BINARY)				Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
				X	5.71	5.71	-6.71	-6.71	-6.71	-1.83	2.20	14.05	26.51	26.51	23.00	18.97	17.61	21.63	26.51	26.51	26.51	14.09	14.09	5.71	-6.71	-6.71	-3.20	0.83
				Y	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55	-2.55
Position	X	Y	Z	Z	-13.35	-8.97	-8.99	-13.35	-17.80	-17.80	-17.80	-17.80	-17.80	-22.18	-26.63	-26.63	-34.36	-34.36	-34.36	-38.81	-43.19	-43.19	-38.81	-34.36	-34.36	-29.98	-25.53	-25.53
					1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	5.71	-2.55	-13.35	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
2	5.71	-2.55	-8.97	2	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	-6.71	-2.55	-8.99	3	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	-6.71	-2.55	-13.35	4	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	-6.71	-2.55	-17.80	5	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
6	-1.83	-2.55	-17.80	6	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	2.20	-2.55	-17.80	7	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	14.05	-2.55	-17.80	8	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
9	26.51	-2.55	-17.80	9	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	26.51	-2.55	-22.18	10	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
11	23.00	-2.55	-26.63	11	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
12	18.97	-2.55	-26.63	12	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
13	17.61	-2.55	-34.36	13	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0
14	21.63	-2.55	-34.36	14	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
15	26.51	-2.55	-34.36	15	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
16	26.51	-2.55	-38.81	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0
17	26.51	-2.55	-43.19	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0
18	14.09	-2.55	-43.19	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0
19	14.09	-2.55	-38.81	19	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0
20	5.71	-2.55	-34.36	20	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
21	-6.71	-2.55	-34.36	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0
22	-6.71	-2.55	-29.98	22	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
23	-3.20	-2.55	-25.53	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
24	0.83	-2.55	-25.53	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 5. Binary Position-to-Rotator Matrix for user input

POSITION TO ROTATOR (BINARY)				Rotator	1	2	3	4	5	6	7	8	9	10	11	12
				X	5.714	-1.827	26.514	18.974	14.090	21.631	-6.711	0.830	14.094	14.094	5.710	5.710
				Y	-2.552	-2.552	-2.552	-2.552	-2.552	-2.552	-2.552	-2.552	-2.552	-2.552	-2.552	-2.552
Position	X	Y	Z	Z	-17.797	-13.352	-26.626	-22.181	-34.365	-38.808	-25.534	-29.979	-22.181	-26.626	-25.534	-29.979
					1	2	3	4	5	6	7	8	9	10	11	12
1	5.714	-2.552	-13.352	1	1	1	0	0	0	0	0	0	0	0	1	1
2	5.714	-2.552	-8.966	2	0	0	0	0	0	0	0	0	0	0	0	0
3	-6.707	-2.552	-8.986	3	0	0	0	0	0	0	0	0	0	0	0	0
4	-6.707	-2.552	-13.352	4	0	1	0	0	0	0	0	0	0	0	0	0
5	-6.707	-2.552	-17.797	5	0	0	0	0	0	0	1	0	0	0	0	0
6	-1.827	-2.552	-17.797	6	0	1	0	0	0	0	0	0	0	0	0	0
7	2.199	-2.552	-17.797	7	1	0	0	0	0	0	0	0	0	0	0	0
8	14.048	-2.552	-17.795	8	1	0	0	0	1	0	0	0	1	1	0	0
9	26.514	-2.552	-17.795	9	0	0	0	0	0	0	0	0	0	0	0	0
10	26.514	-2.552	-22.181	10	0	0	1	1	0	0	0	0	1	0	0	0
11	22.999	-2.552	-26.626	11	0	0	1	0	0	0	0	0	0	0	0	0
12	18.974	-2.552	-26.626	12	0	0	0	1	0	0	0	0	0	1	0	0
13	17.605	-2.552	-34.363	13	0	0	0	0	1	0	0	0	0	0	0	0
14	21.631	-2.552	-34.363	14	0	0	0	0	0	1	0	0	0	0	0	0
15	26.510	-2.552	-34.363	15	0	0	0	0	0	0	0	0	0	0	0	0
16	26.510	-2.552	-38.808	16	0	0	0	0	0	1	0	0	0	0	0	0
17	26.510	-2.552	-43.193	17	0	0	0	0	0	0	0	0	0	0	0	0
18	14.090	-2.552	-43.193	18	0	0	0	0	0	0	0	0	0	0	0	0
19	14.090	-2.552	-38.808	19	0	0	0	0	1	1	0	0	1	1	0	0
20	5.710	-2.552	-34.365	20	1	0	0	0	1	0	0	0	0	0	1	1
21	-6.711	-2.552	-34.365	21	0	0	0	0	0	0	0	0	0	0	0	0
22	-6.711	-2.552	-29.979	22	0	0	0	0	0	0	1	1	0	0	0	1
23	-3.195	-2.552	-25.534	23	0	0	0	0	0	0	1	0	0	0	0	0
24	0.830	-2.552	-25.534	24	0	0	0	0	0	0	0	1	0	0	1	0

Table 6. Binary Rotator-to-Rotator Matrix for user input

ROTATOR TO ROTATOR (BINARY)				Rotator	1	2	3	4	5	6	7	8	9	10	11	12
				X	5.713849	-1.82688	26.51447	18.97371	14.08998	21.63072	-6.7106	0.830146	14.09398	14.09398	5.709882	5.709882
				Y	-2.55176	-2.55176	-2.55176	-2.55176	-2.55176	-2.55176	-2.55176	-2.55176	-2.55176	-2.55176	-2.55176	-2.55176
Rotator	X	Y	Z	Z	-17.7965	-13.3516	-26.626	-22.181	-34.3647	-38.8075	-25.5339	-29.9789	-22.181	-26.626	-25.5339	-29.9789
					1	2	3	4	5	6	7	8	9	10	11	12
1	5.713849	-2.55176	-17.7965	1	0	0	0	0	0	0	0	0	0	0	1	0
2	-1.82688	-2.55176	-13.3516	2	0	0	0	0	0	0	0	0	0	0	0	0
3	26.51447	-2.55176	-26.626	3	0	0	0	0	0	0	0	0	0	0	0	0
4	18.97371	-2.55176	-22.181	4	0	0	0	0	0	0	0	0	1	0	0	0
5	14.08998	-2.55176	-34.3647	5	0	0	0	0	0	0	0	0	0	1	0	0
6	21.63072	-2.55176	-38.8075	6	0	0	0	0	0	0	0	0	0	0	0	0
7	-6.7106	-2.55176	-25.5339	7	0	0	0	0	0	0	0	0	0	0	0	0
8	0.830146	-2.55176	-29.9789	8	0	0	0	0	0	0	0	0	0	0	0	1
9	14.09398	-2.55176	-22.181	9	0	0	0	1	0	0	0	0	0	1	0	0
10	14.09398	-2.55176	-26.626	10	0	0	0	0	1	0	0	0	1	0	0	0
11	5.709882	-2.55176	-25.5339	11	1	0	0	0	0	0	0	0	0	0	0	1
12	5.709882	-2.55176	-29.9789	12	0	0	0	0	0	0	0	1	0	0	1	0

A different methodology is used to add the values that come from the cells that are linked through paths that have rotators in them. A tab named “Connections-W-Rotators” captures the rotators that connect two different cells (up to 3 rotators per link) and calculates the distances from each cell to the rotators and the distances between rotators as well. The user is required to input the identifying number of the first, last and in-between rotators that are part of the path from one work cell to another. In Tables 7 to 10 below, for example, work cell 1 is connected to work cell 6 through rotator 2; in this case rotator 2 is identified as the first rotator and also as the last one. Similarly, work cell 7 is connected to work cell 24 through rotators 1 and 11; rotator 1 is identified as the first one and rotator 11 is identified as the last one. The rest of the input values are displayed in the following tables.

Table 7. Connection with Rotators user input – Section 1

		Positions																														
Rotators From-To		1	1	4	4	5	5	6	6	7	7	8	8	10	10	11	11	12	12	13	13	14	14	16	16	19	19	20	20	22	22	23	23	24	24	
1	1st Rotator?		0		0		0	2	8	1	4	1	4		0		0		0		0		0		0		0		0	12	17		0	11	12	1st Rotator?
1	Last Rotator?		0		0		0	2	4	1	4	1	8		0		0		0		0		0		0		0		0	12	12		0	11	5	Last Rotator?
1	Dist.Between		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	Dist.Between
1	3rd Rotator?		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	3rd Rotator?
1	Total		0		0		0		12		8		13		0		0		0		0		0		0		0		0		29		0		17	Total
4	1st Rotator?		0		0		0	2	5		0		0		0		0		0		0		0		0		0		0		0		0		0	1st Rotator?
4	Last Rotator?		0		0		0	2	4		0		0		0		0		0		0		0		0		0		0		0		0		0	Last Rotator?
4	Dist.Between		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	Dist.Between
4	3rd Rotator?		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	3rd Rotator?
4	Total		0		0		0		9		0		0		0		0		0		0		0		0		0		0		0		0		0	Total
6	1st Rotator?	2	4	2	4		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	1st Rotator?
6	Last Rotator?	2	8	2	5		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	Last Rotator?
6	Dist.Between		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	Dist.Between
6	3rd Rotator?		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	3rd Rotator?
6	Total		12		9		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	Total
7	1st Rotator?	1	4		0		0		0		0		0		0		0		0		0		0		0		0	1	4	1	4		0	1	4	1st Rotator?
7	Last Rotator?	1	4		0		0		0		0		0		0		0		0		0		0		0		0	1	17	12	12		0	11	5	Last Rotator?
7	Dist.Between		0		0		0		0		0		0		0		0		0		0		0		0		0		0		12		0		8	Dist.Between
7	3rd Rotator?		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	3rd Rotator?
7	Total		8		0		0		0		0		0		0		0		0		0		0		0		0		20		28		0		16	Total

Table 8. Connection with Rotators user input – Section 2

		Positions																														
Rotators From-To		1	1	4	4	5	5	6	6	7	7	8	8	10	10	11	11	12	12	13	13	14	14	16	16	19	19	20	20	22	22	23	23	24	24	
8	1st Rotator?	1	8		0		0		0		0		0	9	4		0	10	9	5	17		0		0		0	1	8	1	8		0	1	8	1st Rotator?
8	Last Rotator?	1	4		0		0		0		0		0	9	12		0	10	5	5	4		0		0		0	1	17	12	12		0	11	5	Last Rotator?
8	Dist.Between		0		0		0		0		0		0		0		0		0		0		0		0		0		0		12		0		8	Dist.Between
8	3rd Rotator?		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	3rd Rotator?
8	Total		13		0		0		0		0		0		17		0		14		20		0		0		0		25		33		0		21	Total
10	1st Rotator?		0		0		0		0		0	9	12		0	3	4	4	8	9	12		0		0	9	12	9	12		0		0		0	1st Rotator?
10	Last Rotator?		0		0		0		0		0	9	4		0	3	4	4	4	5	4		0		0	9	17	5	8		0		0		0	Last Rotator?
10	Dist.Between		0		0		0		0		0		0		0		0		0		12		0		0		0		12		0		0		0	Dist.Between
10	3rd Rotator?		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	3rd Rotator?
10	Total		0		0		0		0		0		17		0		8		12		28		0		0		29		33		0		0		0	Total
11	1st Rotator?		0		0		0		0		0		0	3	4		0		0		0		0		0		0		0		0		0		0	1st Rotator?
11	Last Rotator?		0		0		0		0		0		0	3	4		0		0		0		0		0		0		0		0		0		0	Last Rotator?
11	Dist.Between		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	Dist.Between
11	3rd Rotator?		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	3rd Rotator?
11	Total		0		0		0		0		0		0		8		0		0		0		0		0		0		0		0		0		0	Total
12	1st Rotator?		0		0		0		0		0	10	5	4	4		0		0	10	5		0		0	10	5	10	5		0		0		0	1st Rotator?
12	Last Rotator?		0		0		0		0		0	10	9	4	8		0		0	5	4		0		0	10	12	5	8		0		0		0	Last Rotator?
12	Dist.Between		0		0		0		0		0		0		0		0		0		8		0		0		0		8		0		0		0	Dist.Between
12	3rd Rotator?		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	3rd Rotator?
12	Total		0		0		0		0		0		14		12		0		0		16		0		0		17		21		0		0		0	Total

Table 9. Connection with Rotators user input – Section 3

		Positions																														
Rotators From-To		1	1	4	4	5	5	6	6	7	7	8	8	10	10	11	11	12	12	13	13	14	14	16	16	19	19	20	20	22	22	23	23	24	24	
13	1st Rotator?		0		0		0		0		0	5	4	5	4		0	5	4		0		0		0	5	4		0		0		0		0	1st Rotator?
13	Last Rotator?		0		0		0		0		0	5	17	9	12		0	10	5		0		0		0	5	4		0		0		0		0	Last Rotator?
13	Dist.Between		0		0		0		0		0		0		12		0		8		0		0		0		0		0		0		0		0	Dist.Between
13	3rd Rotator?		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	3rd Rotator?
13	Total		0		0		0		0		0		20		28		0		16		0		0		0		8		0		0		0		0	Total
14	1st Rotator?		0		0		0		0		0		0		0		0		0		0		0	6	4	6	4		0		0		0		0	1st Rotator?
14	Last Rotator?		0		0		0		0		0		0		0		0		0		0		0	6	5	6	8		0		0		0		0	Last Rotator?
14	Dist.Between		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	Dist.Between
14	3rd Rotator?		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	3rd Rotator?
14	Total		0		0		0		0		0		0		0		0		0		0		0		9		12		0		0		0		0	Total
16	1st Rotator?		0		0		0		0		0		0		0		0		0		0	6	5		0		0		0		0		0		0	1st Rotator?
16	Last Rotator?		0		0		0		0		0		0		0		0		0		0	6	4		0		0		0		0		0		0	Last Rotator?
16	Dist.Between		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	Dist.Between
16	3rd Rotator?		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	3rd Rotator?
16	Total		0		0		0		0		0		0		0		0		0		0		9		0		0		0		0		0		0	Total
19	1st Rotator?		0		0		0		0		0		0	9	17		0	10	12	5	4	6	8		0		0	5	4		0		0		0	1st Rotator?
19	Last Rotator?		0		0		0		0		0		0	9	12		0	10	5	5	4	6	4		0		0	5	8		0		0		0	Last Rotator?
19	Dist.Between		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	Dist.Between
19	3rd Rotator?		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	3rd Rotator?
19	Total		0		0		0		0		0		0		29		0		17		8		12		0		0		13		0		0		0	Total

Table 10. Connection with Rotators user input – Section 4

		Positions																														
Rotators From-To		1	1	4	4	5	5	6	6	7	7	8	8	10	10	11	11	12	12	13	13	14	14	16	16	19	19	20	20	22	22	23	23	24	24	
20	1st Rotator?		0		0		0		0	1	17	1	17	5	8		0	5	8		0		0		0	5	8		0	12	4		0	12	4	1st Rotator?
20	Last Rotator?		0		0		0		0	1	4	1	8	9	12		0	10	5		0		0		0	5	4		0	12	12		0	8	4	Last Rotator?
20	Dist.Between		0		0		0		0		0		0		12		0		8		0		0		0		0		0		0		0		5	Dist.Between
20	3rd Rotator?		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	3rd Rotator?
20	Total		0		0		0		0		20		25		33		0		21		0		0		0		13		0		17		0		14	Total
22	1st Rotator?	12	12		0		0		0	12	12	12	12		0		0		0		0		0		0		0	12	12		0	7	4	8	8	1st Rotator?
22	Last Rotator?	12	17		0		0		0	1	4	1	8		0		0		0		0		0		0		0	12	4		0	7	4	8	4	Last Rotator?
22	Dist.Between		0		0		0		0		12		12		0		0		0		0		0		0		0		0		0		0		0	Dist.Between
22	3rd Rotator?		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	3rd Rotator?
22	Total		29		0		0		0		28		33		0		0		0		0		0		0		0		17		0		8		12	Total
23	1st Rotator?		0		0	7	4		0		0		0		0		0		0		0		0		0		0		0	7	4		0		0	1st Rotator?
23	Last Rotator?		0		0	7	8		0		0		0		0		0		0		0		0		0		0		0	7	4		0		0	Last Rotator?
23	Dist.Between		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	Dist.Between
23	3rd Rotator?		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	3rd Rotator?
23	Total		0		0	11			0		0		0		0		0		0		0		0		0		0		0		8		0		0	Total
24	1st Rotator?	11	5		0		0		0	11	5	11	5		0		0		0		0		0		0		0	11	5	8	4		0		0	1st Rotator?
24	Last Rotator?	11	12		0		0		0	1	4	1	8		0		0		0		0		0		0		0	11	9	8	8		0		0	Last Rotator?
24	Dist.Between		0		0		0		0		8		8		0		0		0		0		0		0		0		0		0		0		0	Dist.Between
24	3rd Rotator?		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0	3rd Rotator?
24	Total		17		0		0		0		16		21		0		0		0		0		0		0		0		14		12		0		0	Total

The formulas involved in the "Connections-W-Rotators" tab are important to accurately calculate the distances between two work cells that are connected through rotators. If there are rotators listed between two cells, the Excel formulas can calculate the distances between the work cells and between up to three rotators that may be in the path from one cell to another. The formulas used to calculate the distances are displayed in Table 11 below [36-39].

Table 11. Connection with Rotators Excel formulas [36-39]

	Legend	Associated Excel Formula
1st	1st Rotator?	=IF(C3<>0,(INDEX('Rotators-Positions'!$F$6:$Q$29,$A3,C3)*INDEX('Rotators-Positions-Binary'!$F$6:$Q$29,$A3,C3)),0)
2nd	Last Rotator?	=IF(C4<>0,(INDEX('Rotators-Positions'!$F$6:$Q$29,D$2,C4)*INDEX('Rotators-Positions-Binary'!$F$6:$Q$29,D$2,C4)),0)
3rd	Dist.Between	=IF(AND(C3<>0,C4<>0),(IF(C6=0,(INDEX('Rotators-Rotators'!$F$6:$Q$29,C3,C4)*INDEX('Rotators-Rotators-Binary'!$F$6:$Q$29,C3,C4)),(INDEX('Rotators-Rotators'!$F$6:$Q$29,C3,C6)*INDEX('Rotators-Rotators-Binary'!$F$6:$Q$29,C3,C6)))),0)
4th	3rd Rotator?	=IF(C6=0,0,(INDEX('Rotators-Rotators'!$F$6:$Q$29,C4,C6)*INDEX('Rotators-Rotators-Binary'!$F$6:$Q$29,C4,C6)))
5th	Total	=SUM(D3:D6)

If there is only one rotator between two cells like displayed in Table 12, the first Excel formula in Table 11 looks for the distance value listed in the “Rotators-Positions” tab between the first work cell and the rotator. Once it obtains this value, the formula multiplies it by the zero or one that has been assigned in the “Rotators-Positions-Binary” tab. At the same time, the second formula in Table 11 finds the distance and binary values between the rotator and the second work cell in the “Rotators-Positions” and the “Rotators-Positions-Binary” tabs.

If there are two rotators between two cells like displayed in Table 13, the first Excel formula in Table 11 looks for the distance between the first cell and the first rotator. The second Excel formula looks for the distance between the second work cell and the last rotator. Meanwhile, the third Excel formula listed in Table 11 calculates the distance between the two rotators involved.

The formula searches the distance value listed in the "Rotators-Rotators" tab between the two rotators. Then, it multiplies that value by the zero or one that has been assigned in the "Rotators-Rotators-Binary" tab.

Table 12. One rotator only

Rotators From-To		7	7
1	1st Rotator?	1	4
1	Last Rotator?	1	4
1	Dist.Between		0
1	3rd Rotator?		0
1	Total		8

Table 13. Two rotators

Rotators From-To		24	24
7	1st Rotator?	1	4
7	Last Rotator?	11	5
7	Dist.Between		8
7	3rd Rotator?		0
7	Total		16

Although, the layout used for this project does not have three rotators connecting two different work cells, the templates are designed to consider a third rotator if needed. Table 14 below displays a hypothetical example with a third rotator between work cells 7 and 24. With this scenario, the fourth Excel formula in Table 11 calculates the distance between the third rotator and the last rotator (rotator 11 and 12) while the third formula calculates the distance between the first and third rotator (rotator 1 and 12). Simultaneously, the first and second formulas in Table 11 calculate the distances between the first work cell (7) and the first rotator (1) and between the last rotator (11) and the second work cell (24) respectively. Finally, the fifth formula sums up all the calculated values. The total distance between work cells 7 and 24 in this hypothetical example is 25 units.

Table 14. Three rotators if needed

Rotators From-To		24	24
7	1st Rotator?	1	4
7	Last Rotator?	11	5
7	Dist.Between		12
7	3rd Rotator?	12	4
7	Total		25

In this same tab, the totals are consolidated in a work cell-to-work cell matrix by using offset formulas and indirect references. The offset formula consolidates all the totals in a given row in the "Connections-W-Rotators" table. For example, the total values for the row associated with work cell 1 are calculated by a series of formulas like the following: "=OFFSET(BX7,0,BX$6)" [40]. This formula extracts the distance value with rotators between work cell 1 and work cell 24. Table 15 below shows the results of the offset formula for work cell 1. Notice that the second row displays several negative numbers; these numbers are the column pointers to the position from which the value must be extracted. In the OFFSET formula above, the first value in the parenthesis is the cell where the reference will be started from. In this case, the cell is the same one that houses the formula and gets the result: BX7. The second value in the parenthesis is the number of rows up or down where the offset must go; in this case, zero rows are offset. Finally, the third value in the parenthesis is the number of columns to the left or the right where the offset must go; in this case, -26 indicate an offset of 26 columns to the left of the formula cell. The result of this formula example is 17 which is the distance with rotators between work cells 1 and 24. See last column in Table 15.

Table 15. Offset Formula results for work cell 1

	OFFSET																							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	-49	-48	-47	-46	-45	-44	-43	-42	-41	-40	-39	-38	-37	-36	-35	-34	-33	-32	-31	-30	-29	-28	-27	-26
	0	0	0	0	0	12	8	13	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	17

The "=INDIRECT(reference)" formula [41] uses a Excel reference name to extract its value and display it in the cell where the indirect formula is used. To display how this formula works, Table 16 below shows the reference values and Table 17 contains the formula results. The value between work cell 1 and work cell 24 is calculated in Table 17 by using the following formula: "=INDIRECT(CY7)." CY7 is the Excel cell that houses the reference name of BX7 in Table 16.

BX7 is the cell that stores the distance between work cells 1 and 7 that results from the OFFSET formula explained before (Table 15).

Table 16. Excel INDIRECT( ) formula reference values

REFERENCES																								
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	BA7	BB7	BC7	BD7	BE7	BF7	BG7	BH7	BI7	BJ7	BK7	BL7	BM7	BN7	BO7	BP7	BQ7	BR7	BS7	BT7	BU7	BV7	BW7	BX7
2	BA12	BB12	BC12	BD12	BE12	BF12	BG12	BH12	BI12	BJ12	BK12	BL12	BM12	BN12	BO12	BP12	BQ12	BR12	BS12	BT12	BU12	BV12	BW12	BX12
3	BA17	BB17	BC17	BD17	BE17	BF17	BG17	BH17	BI17	BJ17	BK17	BL17	BM17	BN17	BO17	BP17	BQ17	BR17	BS17	BT17	BU17	BV17	BW17	BX17
4	BA22	BB22	BC22	BD22	BE22	BF22	BG22	BH22	BI22	BJ22	BK22	BL22	BM22	BN22	BO22	BP22	BQ22	BR22	BS22	BT22	BU22	BV22	BW22	BX22
5	BA27	BB27	BC27	BD27	BE27	BF27	BG27	BH27	BI27	BJ27	BK27	BL27	BM27	BN27	BO27	BP27	BQ27	BR27	BS27	BT27	BU27	BV27	BW27	BX27
6	BA32	BB32	BC32	BD32	BE32	BF32	BG32	BH32	BI32	BJ32	BK32	BL32	BM32	BN32	BO32	BP32	BQ32	BR32	BS32	BT32	BU32	BV32	BW32	BX32
7	BA37	BB37	BC37	BD37	BE37	BF37	BG37	BH37	BI37	BJ37	BK37	BL37	BM37	BN37	BO37	BP37	BQ37	BR37	BS37	BT37	BU37	BV37	BW37	BX37
8	BA42	BB42	BC42	BD42	BE42	BF42	BG42	BH42	BI42	BJ42	BK42	BL42	BM42	BN42	BO42	BP42	BQ42	BR42	BS42	BT42	BU42	BV42	BW42	BX42
9	BA47	BB47	BC47	BD47	BE47	BF47	BG47	BH47	BI47	BJ47	BK47	BL47	BM47	BN47	BO47	BP47	BQ47	BR47	BS47	BT47	BU47	BV47	BW47	BX47
10	BA52	BB52	BC52	BD52	BE52	BF52	BG52	BH52	BI52	BJ52	BK52	BL52	BM52	BN52	BO52	BP52	BQ52	BR52	BS52	BT52	BU52	BV52	BW52	BX52
11	BA57	BB57	BC57	BD57	BE57	BF57	BG57	BH57	BI57	BJ57	BK57	BL57	BM57	BN57	BO57	BP57	BQ57	BR57	BS57	BT57	BU57	BV57	BW57	BX57
12	BA62	BB62	BC62	BD62	BE62	BF62	BG62	BH62	BI62	BJ62	BK62	BL62	BM62	BN62	BO62	BP62	BQ62	BR62	BS62	BT62	BU62	BV62	BW62	BX62
13	BA67	BB67	BC67	BD67	BE67	BF67	BG67	BH67	BI67	BJ67	BK67	BL67	BM67	BN67	BO67	BP67	BQ67	BR67	BS67	BT67	BU67	BV67	BW67	BX67
14	BA72	BB72	BC72	BD72	BE72	BF72	BG72	BH72	BI72	BJ72	BK72	BL72	BM72	BN72	BO72	BP72	BQ72	BR72	BS72	BT72	BU72	BV72	BW72	BX72
15	BA77	BB77	BC77	BD77	BE77	BF77	BG77	BH77	BI77	BJ77	BK77	BL77	BM77	BN77	BO77	BP77	BQ77	BR77	BS77	BT77	BU77	BV77	BW77	BX77
16	BA82	BB82	BC82	BD82	BE82	BF82	BG82	BH82	BI82	BJ82	BK82	BL82	BM82	BN82	BO82	BP82	BQ82	BR82	BS82	BT82	BU82	BV82	BW82	BX82
17	BA87	BB87	BC87	BD87	BE87	BF87	BG87	BH87	BI87	BJ87	BK87	BL87	BM87	BN87	BO87	BP87	BQ87	BR87	BS87	BT87	BU87	BV87	BW87	BX87
18	BA92	BB92	BC92	BD92	BE92	BF92	BG92	BH92	BI92	BJ92	BK92	BL92	BM92	BN92	BO92	BP92	BQ92	BR92	BS92	BT92	BU92	BV92	BW92	BX92
19	BA97	BB97	BC97	BD97	BE97	BF97	BG97	BH97	BI97	BJ97	BK97	BL97	BM97	BN97	BO97	BP97	BQ97	BR97	BS97	BT97	BU97	BV97	BW97	BX97
20	BA102	BB102	BC102	BD102	BE102	BF102	BG102	BH102	BI102	BJ102	BK102	BL102	BM102	BN102	BO102	BP102	BQ102	BR102	BS102	BT102	BU102	BV102	BW102	BX102
21	BA107	BB107	BC107	BD107	BE107	BF107	BG107	BH107	BI107	BJ107	BK107	BL107	BM107	BN107	BO107	BP107	BQ107	BR107	BS107	BT107	BU107	BV107	BW107	BX107
22	BA112	BB112	BC112	BD112	BE112	BF112	BG112	BH112	BI112	BJ112	BK112	BL112	BM112	BN112	BO112	BP112	BQ112	BR112	BS112	BT112	BU112	BV112	BW112	BX112
23	BA117	BB117	BC117	BD117	BE117	BF117	BG117	BH117	BI117	BJ117	BK117	BL117	BM117	BN117	BO117	BP117	BQ117	BR117	BS117	BT117	BU117	BV117	BW117	BX117
24	BA122	BB122	BC122	BD122	BE122	BF122	BG122	BH122	BI122	BJ122	BK122	BL122	BM122	BN122	BO122	BP122	BQ122	BR122	BS122	BT122	BU122	BV122	BW122	BX122

Table 17. Excel INDIRECT( ) formula results

INDIRECT RESULTS																								
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	0	0	0	0	0	12	8	13	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	17
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	9.3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	12	0	0	9.3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	28	0	16
8	13	0	0	0	0	0	0	0	0	17	0	14	20	0	0	0	0	0	0	25	0	33	0	21
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	17	0	0	8	12	28	0	0	0	0	0	29	33	0	0	0	0
11	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	14	0	12	0	0	16	0	0	0	0	0	17	21	0	0	0	0
13	0	0	0	0	0	0	0	20	0	28	0	16	0	0	0	0	0	0	8	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9.3	0	0	12	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	9.3	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	29	0	17	8	12	0	0	0	0	0	13	0	0	0	0
20	0	0	0	0	0	0	20	25	0	33	0	21	0	0	0	0	0	0	13	0	0	17	0	14
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	29	0	0	0	0	0	28	33	0	0	0	0	0	0	0	0	0	0	0	17	0	0	8	12
23	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0
24	17	0	0	0	0	0	16	21	0	0	0	0	0	0	0	0	0	0	0	14	0	12	0	0

Thanks to these formulas and tables working together, a matrix of 24 by 24 (number of work cells) is created with distances for feasible rotator paths. This matrix can be added to the position-to-position matrix containing distances between cells connected by straight paths. The end result

is a distance matrix that represents all the possible connections in the multi work cell layout and that can be used to find the near optimal route that connects all work cells with Mat Lab. The "Final-Matrix" tab in Excel consolidates and displays this end result. See Table 18 below.

Table 18. Final distance matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	0	4.386	0	12.42	0	11.99	7.96	12.78	0	0	0	0	0	0	0	0	0	0	0	21.01	0	29.05	0	17.06
2	4.386	0	12.42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	12.42	0	4.366	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	12.42	0	4.366	0	4.445	9.325	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	4.445	0	4.88	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12.18	0	0
6	11.99	0	0	9.325	4.88	0	4.025	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	7.96	0	0	0	0	4.025	0	11.85	0	0	0	0	0	0	0	0	0	0	0	20.08	0	28.12	0	16.13
8	12.78	0	0	0	0	0	11.85	0	12.47	16.81	0	13.71	20.08	0	0	0	0	0	21.01	24.9	0	32.94	0	20.95
9	0	0	0	0	0	0	0	12.47	0	4.386	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	16.81	4.386	0	13.63	11.99	28.12	0	0	0	0	0	29.05	32.98	0	0	0	0
11	0	0	0	0	0	0	0	0	0	13.63	0	4.025	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	13.71	0	11.99	4.025	0	16.13	0	0	0	0	0	17.06	21	0	0	0	0
13	0	0	0	0	0	0	0	20.08	0	28.12	0	16.13	0	4.025	0	0	0	0	7.958	11.9	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	4.025	0	4.88	9.325	0	0	11.99	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	4.88	0	4.445	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	9.325	4.445	0	4.386	0	12.42	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4.386	0	12.42	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12.42	0	4.386	0	0	0	0	0
19	0	0	0	0	0	0	0	21.01	0	29.05	0	17.06	7.958	11.99	0	12.42	0	4.386	0	12.82	0	0	0	0
20	21.01	0	0	0	0	0	20.08	24.9	0	32.98	0	21	11.9	0	0	0	0	0	12.82	0	12.42	16.81	0	13.71
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12.42	0	4.386	0	0
22	29.05	0	0	0	12.18	0	28.12	32.94	0	0	0	0	0	0	0	0	0	0	0	16.81	4.386	0	7.96	11.99
23	0	0	0	0	11.25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7.96	0	4.025
24	17.06	0	0	0	0	0	16.13	20.95	0	0	0	0	0	0	0	0	0	0	0	13.71	0	11.99	4.025	0
FINAL DISTANCE MATRIX																								

Lastly, for MatLab algorithm running purposes, the zeros in the Final Distance Matrix are replaced for a larger number, such as 1,000, so that the algorithm does not consider an inexistent link (zeros) as a potential route. The 1,000 is a value significantly higher than the distance value between cells that are connected. This helps MatLab to ignore that value while running its solver. Table 19 below displays the Final Distance Matrix ready for MatLab use.

Table 19. Final distance matrix ready for MatLab use

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	1000	4.386	1000	12.42	1000	11.99	7.96	12.78	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	21.01	1000	29.05	1000	17.06
2	4.386	1000	12.42	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
3	1000	12.42	1000	4.366	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
4	12.42	1000	4.366	1000	4.445	9.325	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
5	1000	1000	1000	4.445	1000	4.88	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	12.18	1000	1000
6	11.99	1000	1000	9.325	4.88	1000	4.025	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
7	7.96	1000	1000	1000	1000	4.025	1000	11.85	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	20.08	1000	28.12	1000	16.13
8	12.78	1000	1000	1000	1000	1000	11.85	1000	12.47	16.81	1000	13.71	20.08	1000	1000	1000	1000	1000	21.01	24.9	1000	32.94	1000	20.95
9	1000	1000	1000	1000	1000	1000	1000	12.47	1000	4.386	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
10	1000	1000	1000	1000	1000	1000	1000	16.81	4.386	1000	13.63	11.99	28.12	1000	1000	1000	1000	1000	29.05	32.98	1000	1000	1000	1000
11	1000	1000	1000	1000	1000	1000	1000	1000	1000	13.63	1000	4.025	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
12	1000	1000	1000	1000	1000	1000	1000	13.71	1000	11.99	4.025	1000	16.13	1000	1000	1000	1000	1000	17.06	21	1000	1000	1000	1000
13	1000	1000	1000	1000	1000	1000	1000	20.08	1000	28.12	1000	16.13	1000	4.025	1000	1000	1000	1000	7.958	11.9	1000	1000	1000	1000
14	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	4.025	1000	4.88	9.325	1000	1000	11.99	1000	1000	1000	1000	1000
15	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	4.88	1000	4.445	1000	1000	1000	1000	1000	1000	1000	1000
16	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	9.325	4.445	1000	4.386	1000	12.42	1000	1000	1000	1000	1000
17	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	4.386	1000	12.42	1000	1000	1000	1000	1000	1000
18	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	12.42	1000	4.386	1000	1000	1000	1000	1000
19	1000	1000	1000	1000	1000	1000	1000	21.01	1000	29.05	1000	17.06	7.958	11.99	1000	12.42	1000	4.386	1000	12.82	1000	1000	1000	1000
20	21.01	1000	1000	1000	1000	1000	20.08	24.9	1000	32.98	1000	21	11.9	1000	1000	1000	1000	1000	12.82	1000	12.42	16.81	1000	13.71
21	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	12.42	1000	4.386	1000	1000
22	29.05	1000	1000	1000	12.18	1000	28.12	32.94	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	16.81	4.386	1000	7.96	11.99
23	1000	1000	1000	1000	11.25	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	7.96	1000	4.025
24	17.06	1000	1000	1000	1000	1000	16.13	20.95	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	13.71	1000	11.99	4.025	1000
FINAL DISTANCE MATRIX																								

3.6 MatLab sequence generation with Genetic Algorithms

The near optimal sequence to be simulated in Unity is found by using two MatLab environments: the Standard Command Window, displayed in Figure 4, and the Global Optimization Tool Box, displayed in Figure 5.

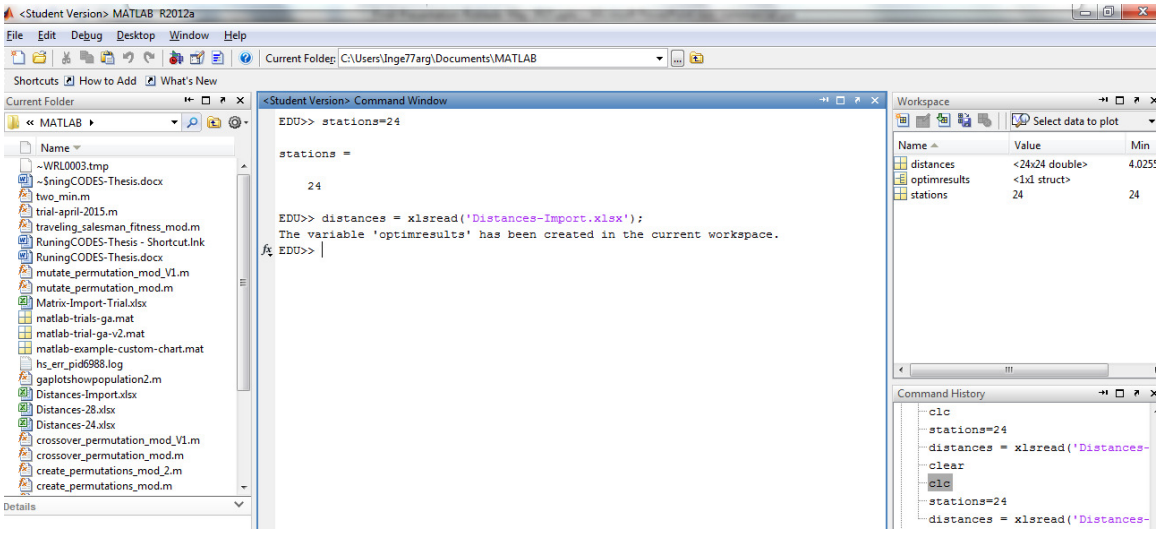

Figure 4. MatLab's Standard command window

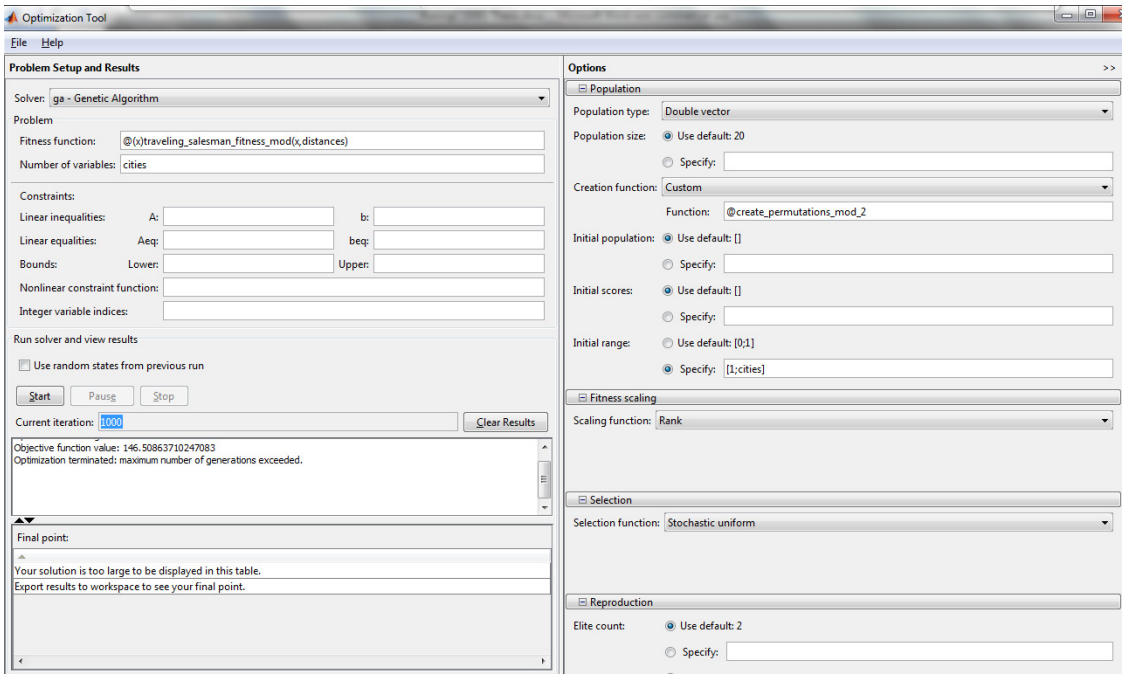

Figure 5. MatLab's Global Optimization Tool Box

In the Standard Command Window, the user must define the number of micro assembly cells that will be used in the simulated virtual reality environment and also must read the distance matrix from the Excel file described in the previous section. The code to accomplish these two things reads as [42]:

EDU>> stations = 24;

EDU>> distances = xlsread('Distances-Import.xlsx');

Once these two parameters are known by MatLab, the Global Optimization tool can be used to find near optimal sequences using genetic algorithms. But before proceeding to present the results for our proposed layout, let me further explain the method that the Global Optimization Tool uses to solve problems with genetic algorithms. The script grabs a sample of permutation sequences and compares them based on the value that they yield to the objective function (fitness value); then, it keeps the top best performers (elite count), applies mutation and crossover operations to them to create a different sample of the population and test it again against the objective function.

Over time, only the best performing individuals (sequences) remain in the group that is being tested for fitness (objective function). When the algorithm stops, the near optimal solution is the sequence that yields the lowest fitness value when tested against the objective function [43].

Our proposed layout has 24 cells; however, let's assume that we have a system that is composed of 70 cells instead and that we still need to supply all of the cells with material. If we modify the Excel distance Matrix to have a size of 70X70 and we upload it to MatLab we can run the genetic algorithm solver for the 70 cell system. See Table 20 below with the hypothetical matrix. When the Global Optimization Tool applies genetic algorithms to find the best individual a plot that displays how the fitness of the generations improves over time can be generated. The plot for the 70 cell layout is shown in Figure 6 below. As it can be seen, the first samples of the population yield fitness values around 280. However, newer and newer generations started displaying better and better fitness values until they reached the near optimal function value of 146.5. The black line in the plot represents the best sequence fitness value that is found over time by the generic algorithm. The blue dots represent fitness values for sequences that are tested by the genetic algorithm. Notice how the blue dots range, which depends on the values in the matrix, decreases and stabilizes when the algorithm approaches the best fitness values.

Table 20. 70X70 distance matrix use for hypothetical example

70x70	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	...	70
1	15	4.386	15	15	15	15	12.42	15	4.386	15	15	15	15	12.42	15	4.386	15	15	15	15	12.42	15	4.386	15	15	12.42
2	4.386	15	7.96	11.99	15	12.42	15	4.386	15	7.96	11.99	15	12.42	15	4.386	15	7.96	11.99	15	12.42	15	4.386	15	7.96	11.99	15
3	15	7.96	15	4.025	15	15	15	15	7.96	15	4.025	15	15	15	15	7.96	15	4.025	15	15	15	15	7.96	15	4.025	15
4	15	11.99	4.025	15	4.88	9.325	15	15	11.99	4.025	15	4.88	9.325	15	15	11.99	4.025	15	4.88	9.325	15	15	11.99	4.025	15	15
5	15	15	15	4.88	15	4.445	15	15	15	15	4.88	15	4.445	15	15	15	15	4.88	15	4.445	15	15	15	15	4.88	15
6	15	12.42	15	9.325	4.445	15	4.366	15	12.42	15	9.325	4.445	15	4.366	15	12.42	15	9.325	4.445	15	4.366	15	12.42	15	9.325	4.37
7	12.42	15	15	15	15	4.366	15	12.42	15	15	15	15	4.366	15	12.42	15	15	15	15	4.366	15	12.42	15	15	15	15
8	15	4.386	15	15	15	15	12.42	15	4.386	15	15	15	15	12.42	15	4.386	15	15	15	15	12.42	15	4.386	15	15	12.42
9	4.386	15	7.96	11.99	15	12.42	15	4.386	15	7.96	11.99	15	12.42	15	4.386	15	7.96	11.99	15	12.42	15	4.386	15	7.96	11.99	15
10	15	7.96	15	4.025	15	15	15	15	7.96	15	4.025	15	15	15	15	7.96	15	4.025	15	15	15	15	7.96	15	4.025	15
11	15	11.99	4.025	15	4.88	9.325	15	15	11.99	4.025	15	4.88	9.325	15	15	11.99	4.025	15	4.88	9.325	15	15	11.99	4.025	15	15
12	15	15	15	4.88	15	4.445	15	15	15	15	4.88	15	4.445	15	15	15	15	4.88	15	4.445	15	15	15	15	4.88	15
13	15	12.42	15	9.325	4.445	15	4.366	15	12.42	15	9.325	4.445	15	4.366	15	12.42	15	9.325	4.445	15	4.366	15	12.42	15	9.325	4.37
14	12.42	15	15	15	15	4.366	15	12.42	15	15	15	15	4.366	15	12.42	15	15	15	15	4.366	15	12.42	15	15	15	15
15	15	4.386	15	15	15	15	12.42	15	4.386	15	15	15	15	12.42	15	4.386	15	15	15	15	12.42	15	4.386	15	15	12.42
16	4.386	15	7.96	11.99	15	12.42	15	4.386	15	7.96	11.99	15	12.42	15	4.386	15	7.96	11.99	15	12.42	15	4.386	15	7.96	11.99	15
17	15	7.96	15	4.025	15	15	15	15	7.96	15	4.025	15	15	15	15	7.96	15	4.025	15	15	15	15	7.96	15	4.025	15
18	15	11.99	4.025	15	4.88	9.325	15	15	11.99	4.025	15	4.88	9.325	15	15	11.99	4.025	15	4.88	9.325	15	15	11.99	4.025	15	15
19	15	15	15	4.88	15	4.445	15	15	15	15	4.88	15	4.445	15	15	15	15	4.88	15	4.445	15	15	15	15	4.88	15
20	15	12.42	15	9.325	4.445	15	4.366	15	12.42	15	9.325	4.445	15	4.366	15	12.42	15	9.325	4.445	15	4.366	15	12.42	15	9.325	4.37
21	12.42	15	15	15	15	4.366	15	12.42	15	15	15	15	4.366	15	12.42	15	15	15	15	4.366	15	12.42	15	15	15	15
22	15	4.386	15	15	15	15	12.42	15	4.386	15	15	15	15	12.42	15	4.386	15	15	15	15	12.42	15	4.386	15	15	12.42
23	4.386	15	7.96	11.99	15	12.42	15	4.386	15	7.96	11.99	15	12.42	15	4.386	15	7.96	11.99	15	12.42	15	4.386	15	7.96	11.99	15
24	15	7.96	15	4.025	15	15	15	15	7.96	15	4.025	15	15	15	15	7.96	15	4.025	15	15	15	15	7.96	15	4.03	15
...	15	11.99	4.025	15	4.88	9.325	15	15	11.99	4.025	15	4.88	9.325	15	15	11.99	4.025	15	4.88	9.325	15	15	11.99	4.03	15.00	15
70	12.42	15	15	15	15	4.366	15	12.42	15	15	15	15	4.366	15	12.42	15	15	15	15	4.366	15	12.42	15	15	15	15

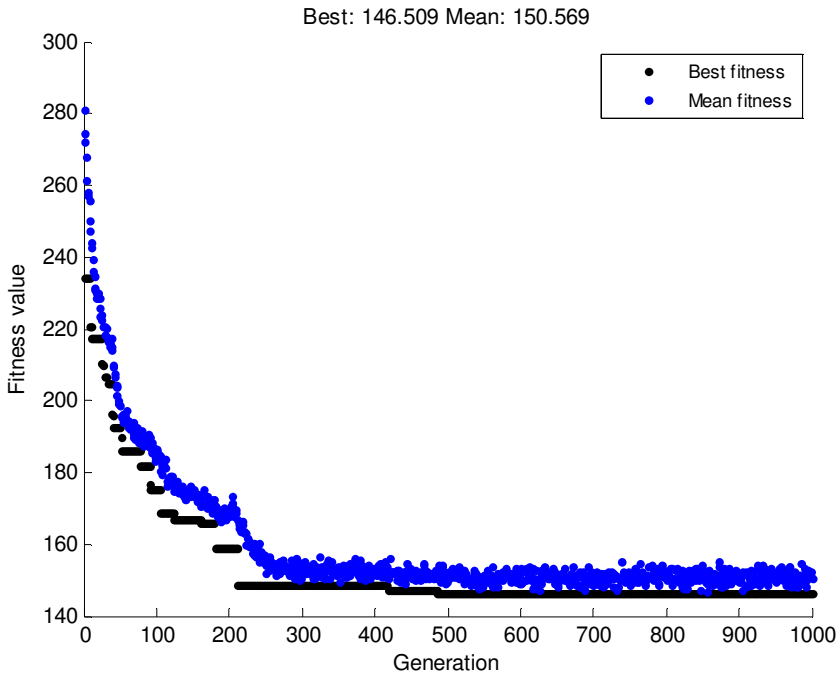

Figure 6. Fitness value plot for 70X70 cell hypothetical example

Returning to our example, the near optimal fitness value for our layout is 184.5829 from the sequence 1 24 23 22 21 20 13 14 15 16 17 18 19 12 11 10 9 8 7 6 5 4 3 2 1. The fitness plot for 1000 generations is displayed in Figure 7 below.

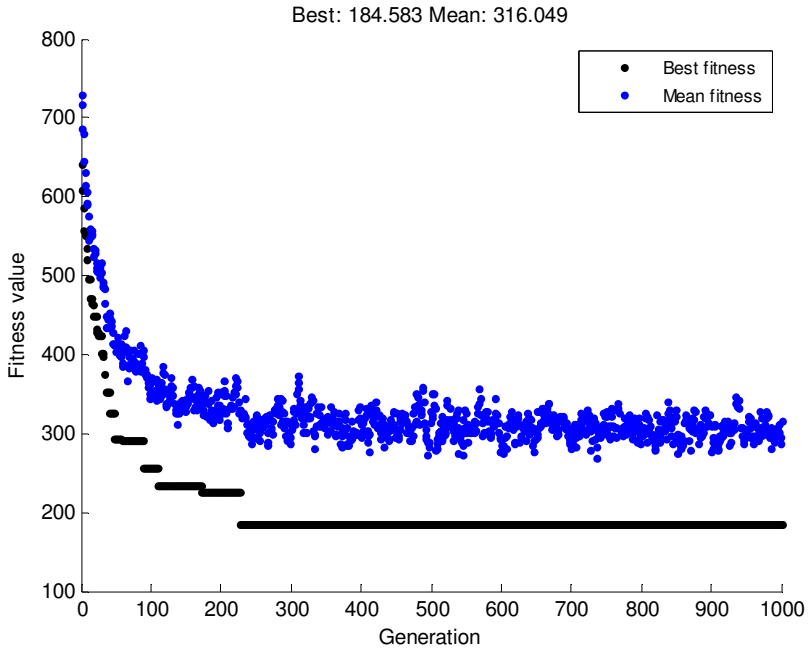

Figure 7. Fitness plot for the 24 work cell layout

At this point in time, we understand how it is that the near optimal sequence is displayed in MatLab once the genetic algorithms are applied; however, you may be still wondering what is used by the Global Optimization Tool to apply the genetic algorithms to the sequence population. The answer is simple: more programming. A permutation generation program is used to create the population of sequences; another program mutates the sequences in the samples being compared; other program applies crossover operators to them. Finally, a program is linked to the Global Optimization Tool to calculate the objective function fitness values. All of these programs and their functions are explained in the following paragraphs.

Since the main objective of this work is the development of a virtual factory environment in Unity to simulate the sequences of material flow in micro assembly factory environments and not the creation of new optimization techniques or programming codes, the scripts used for this project have been based on the programs that MatLab originally uses to solve the salesman traveling problem with genetic algorithms. However, some modifications in the logic had to be applied in order to solve the problem of distributing materials to all the assembly cells in a given micro factory layout.

The "Create_Permutations" script available in [44] and in the appendix is the MatLab code used to generate permutations for the traveling salesman problem. The script generates random sequences (permutations) of "cities" or positions that are visited while an objective function, part of other script, cumulatively adds the distances between the positions that are in the sequence. The permutations created by this script are opened ended and may start and finish their sequence at any random available position. For example, the permutation may start at position one and finish at position ten, twenty or X_i, where "X" is the final position and "i" is the total number of positions. These positions are the work cells or stations in the micro factory context.

In this project it is necessary that the pin pallet starts and stops at the same station. Thus, different script lines are required to fulfill this need. The script code provided below generates permutations that always start and end on the defined initial station (closed loop permutation). The comments after the "%" symbol provide details of what each code line does. Script lines are also available in the appendix.

```
pop = cell(totalPopulationSize,1);%array with size equal to population size
pop2 = cell(totalPopulationSize,1);%array with size equal to population size
for i = 1:totalPopulationSize  %cycle from 1 to population size
    pop2{i} = randperm(n-1)+1; %fills permutations with all stations but station # 1; in an example with 7 cells-it goes from 2 to 7
    for j= 1:(n-1)   %cycle from 1 to number of stations(n)-1
        pop{i}(1,j+1)=pop2{i}(1,j); %starting in position 2 (j=1+1) pop fills with all stations (stored in pop2) except #1
    end
    pop{i,1}(1,1)=1 ;              % all positions #1 are equal to station #1
    pop{i,1}(1,n+1)=pop{i,1}(1,1);     % added last position equal to the first position (1)
end
```

The "Mutate_permutation" script available in [44] and in the appendix is the MatLab code used to mutate, previously created, permutations while the genetic algorithm solver is running. The script randomly changes any positions within the permutation. These random changes do not respect the initial position of the sequence and any of the X_i available positions may end up in the first and last step of the sequence.

Mutations, in the context of this project, are only needed for the positions that are between the first and last steps of the sequence. The value in the first and last positions in the closed loop sequence must be respected to ensure that the material will always start moving from the designated initial work cell and will finish its path there as well. Since the custom "create permutations" script lines described above define an initial station and add it as the last station in the closed loop permutation as well, a code line to remove the last position on each parent that is mutated is needed before the mutation operations are performed. Then, a script line is needed to perform mutation while respecting the first position. Lastly, another script line is needed to reinsert the last position, which is equal to the initial one, after mutation has been applied to generate children sequences. These script lines can be seen below; the comment after the "%" symbol provides details of what the code line does. Script is also available in the appendix.

- *parent(:,n+1)=[];% removes the added last position (by create_permutations_mod) that moves the target back to the initial position at the end of the sequence*
- *p = ceil((length(parent)) * ((1-(1/length(parent)))*rand (1,2)+(1/length(parent))));%ensures that first station stays at beginning and end. Selects a position (station, other than the first one (and last one) to cross over/mutate. Based on the formula to generate values from the uniform distribution on the interval [a, b]:: r = a + (b-a).*rand().*
- *mutationChildren{i,1}(1,n+1)= mutationChildren{i,1}(1,1);% adds to each child sequence one last position with the initial station (1)where the target returns at the end of the sequence*

The "*((1-(1/length(parent)))*rand(1,2)+(1/length(parent)))")* expression is based on the formula to generate values from the uniform distribution on the interval [a,b] where random number "r" is generated by the following expression: r = a + (b-a).*rand; where b=1 and a=(1/length(parent)) which would skip the very first position in a sequence of the parent to be mutated [45, 46].

The "Crossover_permutation" script available at [44] and in the appendix is the MatLab code used to apply crossover operations to the, previously created, permutations while the genetic algorithm solver is running. The script randomly changes any positions within the permutation. These random changes do not respect the initial position of the sequence and any of the X_i available positions may end up in the first and last step of the sequence.

However, just like we discussed above for the mutation script, the crossover operations are only needed for the positions that are between the first and last steps of the sequence. Likewise, a script line to remove the last position on each parent before crossover is needed. Then, a script line is needed to perform crossover while respecting the first position. Finally, a third script line is needed to reinsert the last position in the sequence after crossover operations have been applied to generate children sequences. These script lines can be seen below; the comment after the "%" symbol provides details of what the code line does. Script is also available in the appendix.

- *parent(:,n+1)=[];% removes the added last position (by create_permutations_mod) that moves the target back to the initial position at the end of the sequence*
- *p1 = ceil((length(parent) -1) * ((1-(1/(length(parent)-1)))*rand+ (1/(length(parent)-1)))); %ensures that first station stays at beginning and end. Selects a position (station, other than the first one (and last one) to cross over/mutate. %Based on the formula to generate values from the uniform distribution on the interval [a,b]:: r = a + (b-a).*rand().*
- *xoverKids{i,1}(1,n+1)= xoverKids{i,1}(1,1);% adds to each child sequence one last position with the initial station (1) where the target returns at the end of the sequence*

The second line in the code above is also based on the formula to generate values from the uniform distribution on the interval [a,b] as explained for the mutation script lines above [45, 46].

Finally, the script in MatLab that estimates the objective function value or fitness value is also available at [44] and in the appendix. In this case, however, the script to solve the micro assembly problem is essentially the same but applied in a different context and with different terminology. The only difference is that the distance counting variable "f" must be initialized with a zero *(f = 0;)*which ensures that the only values cumulatively added are the ones of those segments travelled by the target once the sequence is started.

Each one of these scripts are called from the Global Optimization Tool in MatLab before running the Genetic Algorithms. In the following paragraphs we explain each of the Global Optimization Tool sections and the inputs used to call for the appropriate scripts when running the genetic algorithm solver.

In the Problem Setup and Results section the "ga-Genetic Algorithm" is selected as the Solver. Similarly, under the Problem heading, the Fitness Function is selected as "@(x)traveling_salesman_fitness_mod(x,distances)" with number of variables as "stations" [44]. These inputs are displayed in the Figure 8 below.

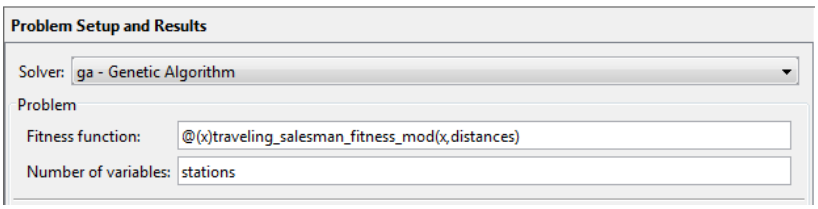

Figure 8. Global Optimization Tool: problem setup and results section

In the Options section, under the Population heading, the creation function is defined as the modified-create-permutations function that is previously discussed: "@create_permutations_mod_2." The initial range is defined as follows: "[1;stations]" [44]. Figure 9 displays these inputs below.

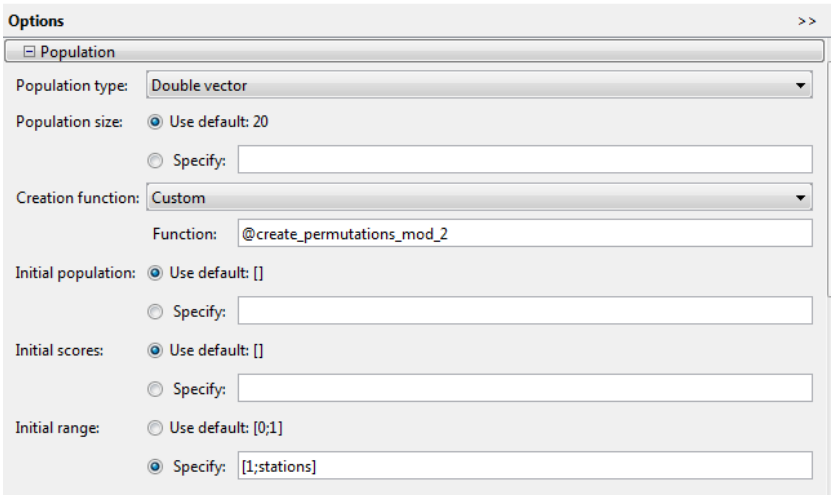

Figure 9. Global Optimization Tool: options section

Under the Reproduction heading, the default value of 2 is used for the elite count and the default value of 0.8 is used for the crossover fraction. The custom function "@mutate_permutation_mod" is called under the Mutation heading and the "@crossover_permutation_mod" function is called under the Crossover heading [44]. These inputs are displayed in Figure 10 below.

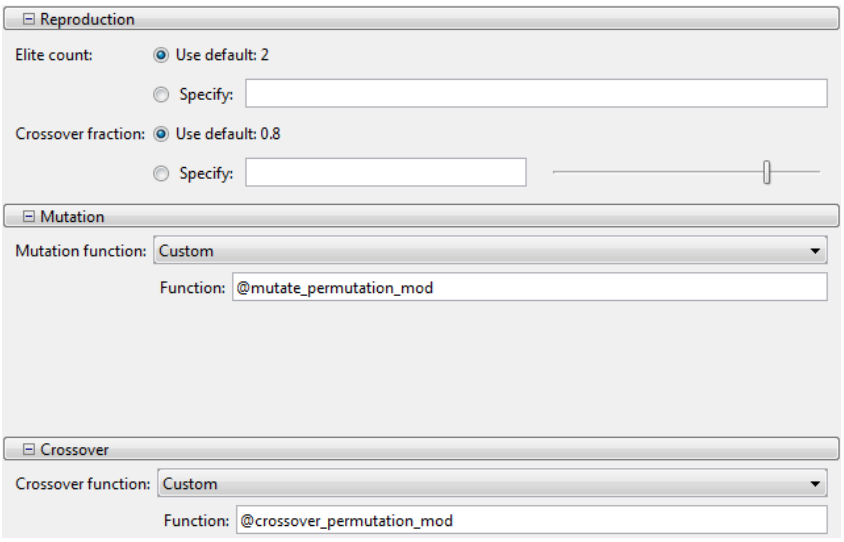

Figure 10. Global Optimization Tool: reproduction section

Finally, under the Stopping criteria heading, the number of generations are set to the desired number and the Best Fitness plot is selected under the Plot functions heading if a fitness value plot is needed.

Once all the sections have the appropriate inputs, the user must use the “Start” button to run the solver. The bottom of the Problem Setup and Results section will automatically display the status of the MatLab run and a pop up window will start creating an fitness value plot that shows the progression of the solutions found by the genetic algorithms as explain earlier in this section.

When the algorithm run is complete, the results section displays the near optimal fitness value and the near optimal sequence is made available to be downloaded into the Standard Command Window. Figure 11 displays the outcome section below.

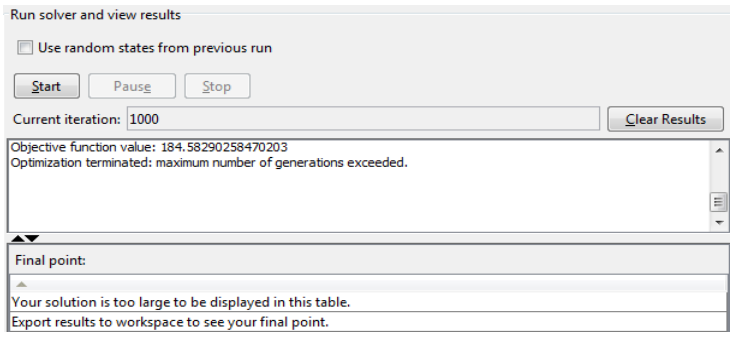

Figure 11. Global Optimization Tool: results section

The user must go to the File menu and then select Export. A pop up window will ask the user what to download. At minimum, as displayed in Figure 12 below, the near optimal sequence values must be downloaded.

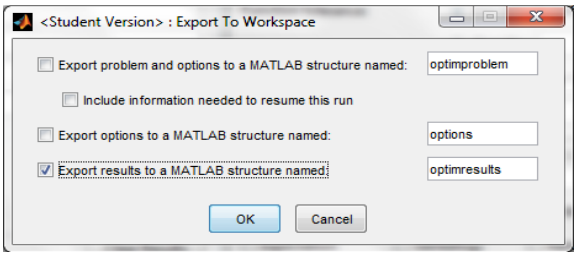

Figure 12. Global Optimization Tool: Export to Workspace window

Once the download is complete, the user double clicks an item under the Workspace window to open the Variable Editor which displays the values of the near optimal sequence as seen in Figure 13 below. This is the sequence that will be simulated in the Virtual Unity environment. The simulation process is explained in the following section.

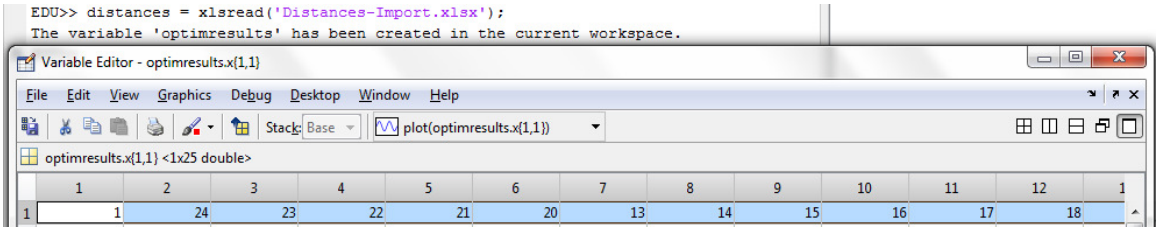

Figure 13. MatLab's variable editor with near optimal sequence displayed

3.7 Micro Assembly Factory Model in Unity ® and Simulation of the near optimal sequence.

3.7.1 Micro Assembly Factory Model

Before being able to simulate near optimal material movement sequences in the micro factory environment it is necessary to build a model in the computer. This project built such a model using software called Unity ® that is commonly used to construct video games.

More detailed Unity model construction information will be provided in next chapter. In this section there will be discussion only on the configuration of the layout used for this project and how Unity can be used or modified to simulate different scenarios that display distribution sequences within the micro assembly factories that are created.

A micro assembly factory with twenty four (24) stations is created to work on this project. See Figure 14. The intention of this project is to use virtual reality software to represent and improve environments that are difficult to work with in the real world due to diverse restrictions such as size or accessibility. Unity enables the user to zoom into the smallest access areas and it also enables the user to animate or simulate motion with scripts. See Figure 15. This micro assembly factory environment was created with help of the resources and full support made available by the Center for Information Centric Engineering and its staff which pertain to the Industrial Engineering and Management department at Oklahoma State University [47].

In our model, there is a pallet that is used to distribute material (pins) throughout the micro factory. The pallet Unity object contains a script that houses the near optimal sequence that it must follow to deliver material to all stations. In the main model conveyors are used to distribute the material; however, different scenarios can be explored. For example, if the stations could be linked after the near optimal routes are found, the Unity model can be used to simulate a sequence

that is not restricted by conveyors. This approach can be used to design micro factories that are more efficient even before they are constructed. This scenario will be further discussed in the alternative layouts section below.

Another scenario would be the one where not all stations need to be serviced. For example, if out of twenty four stations only twelve are in use, our methodology can be used to find the near optimal sequence for twelve stations and to simulate it in the Unity environment.

The most valuable contributions from this project is the creation of the Unity environment and an improvement procedure that enables designers of new micro factories to test different options and select the most convenient while also allows for studying and simulating different scenarios of existing factories.

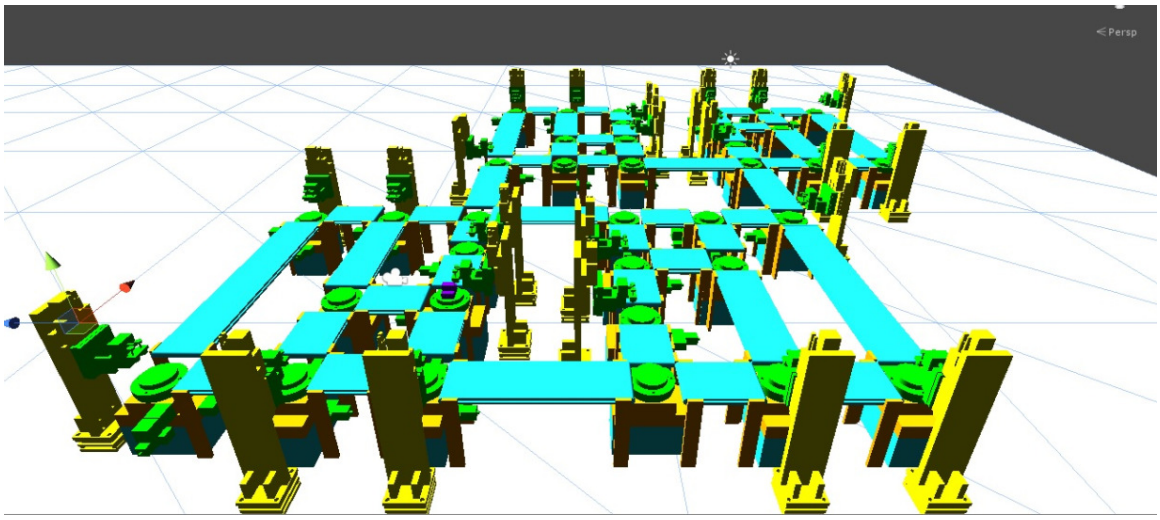

Figure 14. Twenty four work cell layout in Unity®

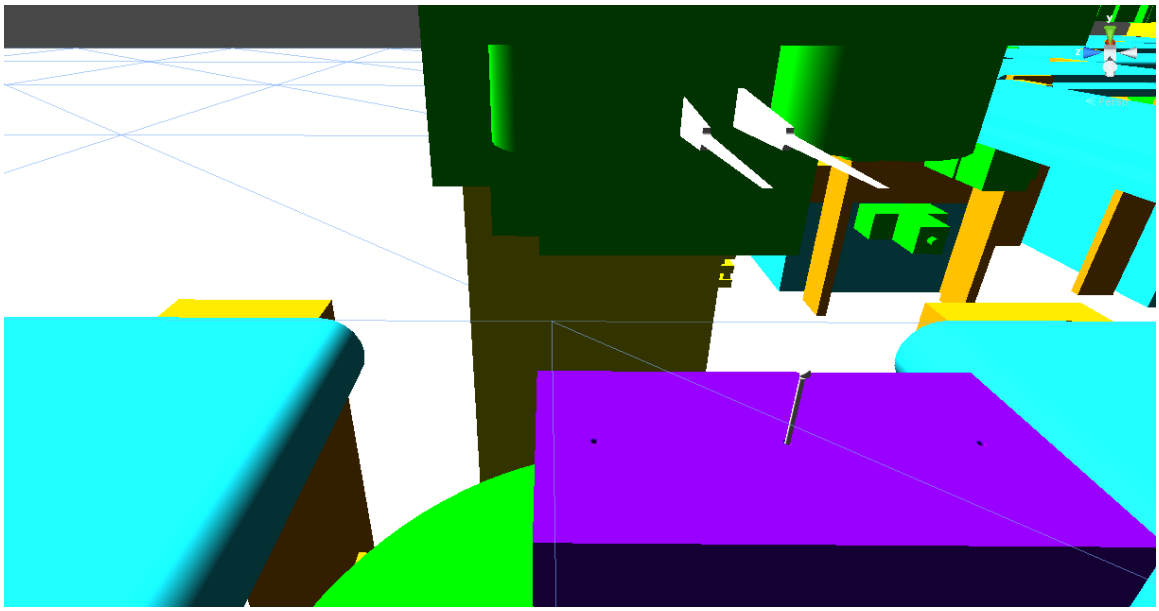

Figure 15. Zoom in to see small pin in Unity®

3.7.2 Simulation of near optimal sequence

In Unity, all items used to represent an environment are known as assets or objects. In our scenario we are using a semi cubic shaped “pallet” to deliver pins to each of the micro assembly stations [29].

In order to simulate the motion of this object within the factory environment, a script needs to be created and attached to the object itself. The script that is attached to our “pallet” in Unity is the one below. Script is also provided in the appendix.

1. *import System; //Initializing code*
2. *import System.IO; //Initializing code*
3. *var target : Transform; //object to be moved*
4. *var Initial: Transform; //object to be the initial place from where the target object starts moving*
5. *var Sequence: Transform[]= new Transform[31]; // array that represents the sequence of stations where target moves*
6. *var speed : float; //Variable that defines how fast the target moves*
7. *var i = 0; //Counter for the "for" cycle that moves the target from one station to another*
8. *var vect: Vector3; //vector definition*

```
9.  private var dist : float; //Variable used to calculate the distance traveled on each
    sequence segment
10. private var Totdist : float;//Variable used to calculate the overall cumulative distance
11. target.position = Initial.position; //Moves target to the initial position
12. print("Initial coordinates: "+Initial.position);//Prints initial position coordinates in the
    Console window
1.  for (i=0; i < Sequence.Length; i++)//cycle that moves target until sequence is completed
2.  {
3.  dist = Vector3.Distance(Sequence[i].position,target.position);//Calculates segment
    distance traveled by target
4.  Totdist=Totdist+dist;//Calculates cumulative traveled distance
5.  vect = Sequence[i].position; //Redefines vect as next (i) sequence position
6.  yield WaitForSeconds((dist/speed)+1);//Introduces a pause for target to stop at each
    station in sequence
7.  print("New coordinates: "+target.position); //Prints coordinates for each station where
    the target stops by
8.  print("Segment distance: "+dist);      //Prints segment distance value
9.  print("Cumulative distance: "+Totdist);//Prints cumulative distance value
10. print(vect);//Prints coordinates of sequence object where target arrives into
11. }
12. function FixedUpdate() {//Function that constantly updates the status of the model (and
    target)
13. target.position = //New position of target equals:
14. Vector3(Mathf.MoveTowards(target.position.x,vect.x,speed*Time.deltaTime),//moving in
    X at speed rate
15. Mathf.MoveTowards(target.position.y,vect.y,speed*Time.deltaTime), //moving in Y at a
    speed rate
16. Mathf.MoveTowards(target.position.z,vect.z,speed*Time.deltaTime));//moving in Z at a
    speed rate
17. }
```

The details of how the script works will be given in next chapter. For now, we will only highlight that this script calls for a sequence input from the user for the pallet to move from place to place. The sequence is input by dragging or listing the other objects (stations) where the pallet must visit next. The sequences input for our project are those ones obtained from the genetic algorithm ran in MatLab. Earlier in this chapter, the User Interface section (3.3.1) discussed how the user inputs the sequence into the pallet script. Figure 16 displays how the script creates a sequence list to be input in the Inspector window when the pallet object is highlighted in the Hierarchy window. The same script also enables the user to define which will be the initial station and how many steps the sequence list has (size field).

Once the sequence objects are listed, the initial station is selected and the number of moves has been defined, the user can simulate the sequence by using the “play” button. The Game screen is displayed immediately and the pallet moves from station to station. Simultaneously, under the console tab the script prints the last segment distance and the cumulative distance travelled by the pallet during the simulation. Figure 17 below displays these prints.

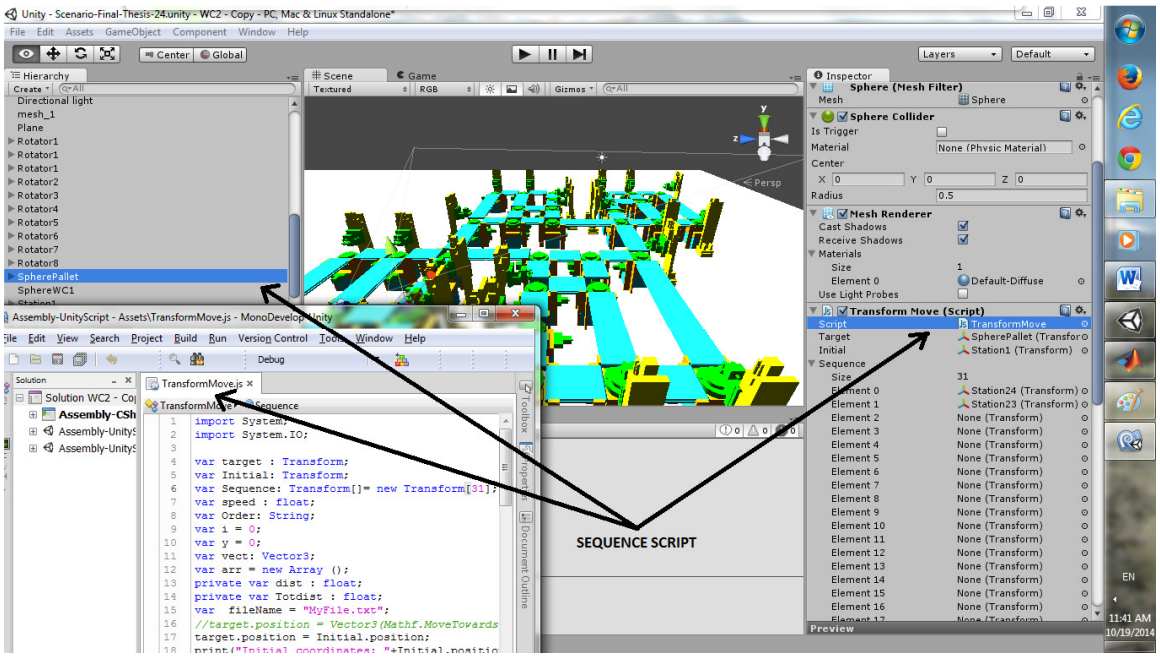

Figure 16. Sequence object list used for user input

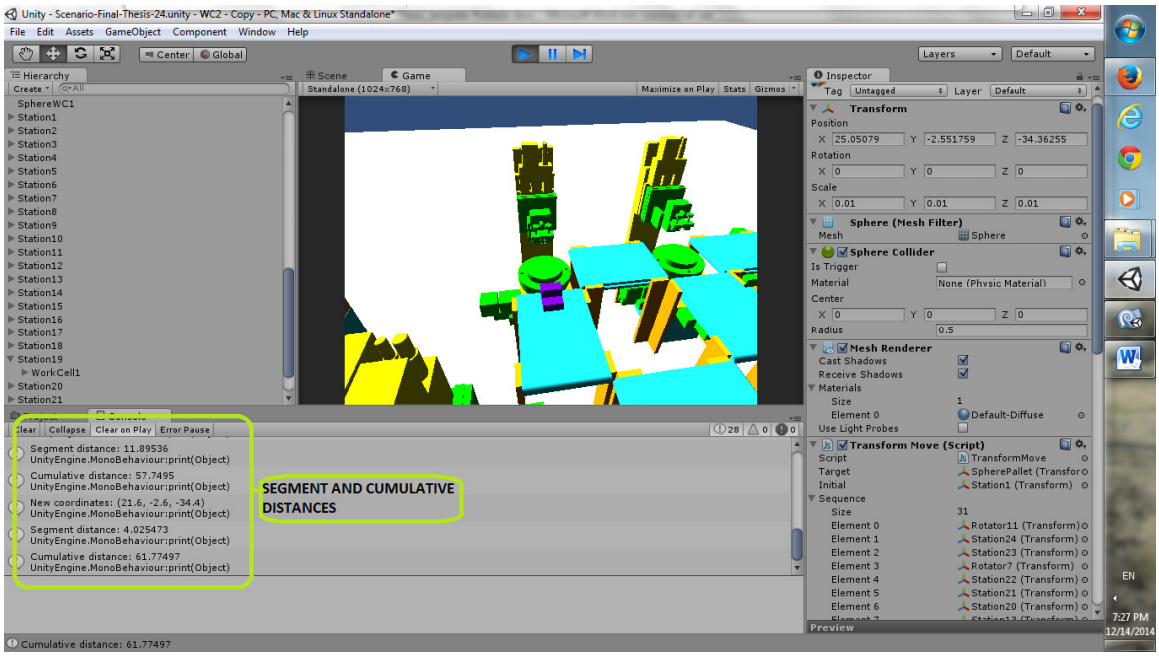

Figure 17. Simulation prints with segment and cumulative distances

The model simulates the pallet visit to the stations 1 24 23 22 21 20 13 14 15 16 17 18 19 12 11 10 9 8 7 6 5 4 3 2 1. The cumulative distance displayed at the end of the sequence is 184.5829 just like it is calculated in the MatLab results. This validates the solution found in MatLab and the accuracy of the methodology proposed by this project to simulate micro assembly factory scenarios.

When different layouts are tested, the cumulative distances calculated by Unity go along with the near optimal sequences generated by the MatLab genetic algorithm. This is further explained in the following section where other layouts and scenarios are explored.

3.8 Alternative Layouts

As previously discussed, the Unity environment can be used to simulate different micro assembly cell scenarios to either better operate existing systems or design future micro factories.

One scenario that can be simulated following the methodology proposed by this project is the replenishment of material to only a portion of the 24 available micro assembly stations. For example, if only 12 stations are supplied, the distance matrix can look like the one below, where the stations available will be only: 1 2 3 4 5 6 7 20 21 22 23 24.

Table 21. Distance Matrix for only twelve Stations

	1	2	3	4	5	6	7	20	21	22	23	24
1	35	4.386	35	12.42	35	11.99	7.96	21.01	35	29.05	35	17.06
2	4.386	35	12.42	35	35	35	35	35	35	35	35	35
3	35	12.42	35	4.386	35	35	35	35	35	35	35	35
4	12.42	35	4.386	35	4.445	9.325	35	35	35	35	35	35
5	35	35	35	4.445	35	4.88	35	35	35	12.18	35	35
6	11.99	35	35	9.325	4.88	35	4.025	35	35	35	35	35
7	7.96	35	35	35	35	4.025	35	20.08	35	28.12	35	16.13
20	21.01	35	35	35	35	35	20.08	35	12.42	16.81	35	13.71
21	35	35	35	35	35	35	35	12.42	35	4.386	35	35
22	29.05	35	35	35	12.18	35	28.12	16.81	4.386	35	7.96	11.99
23	35	35	35	35	11.25	35	35	35	35	7.96	35	4.025
24	17.06	35	35	35	35	35	16.13	13.71	35	11.99	4.025	35
FINAL DISTANCE MATRIX -12 Stations												

When the proposed methodology is applied for this scenario, the MatLab near optimal sequence is: 1 24 23 22 21 20 7 6 5 4 3 2 1. This sequence yields a cumulative pallet travel distance of 100.4798 as shown in Figure 18 below.

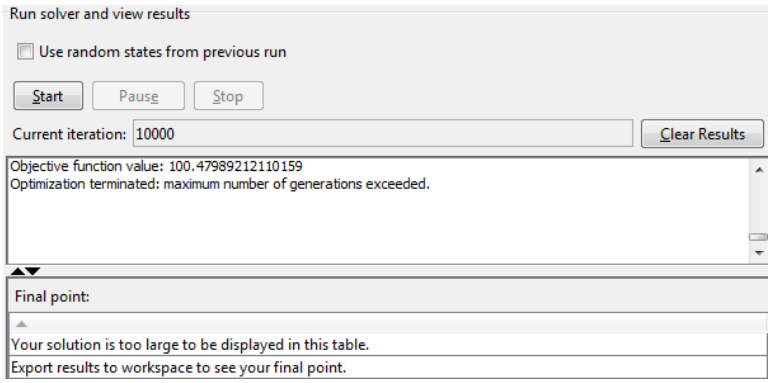

Figure 18. MatLab's near optimal fitness value for the twelve station layout

This sequence is simulated in the Unity environment and the near optimal cumulative value is validated as displayed in Figure 19 below. The twelve station sequence can be seen to the right of the figure while the final cumulative distance value can be seen in the lower left corner. Notice that this value is 100.479 which matches the value obtained in MatLab.

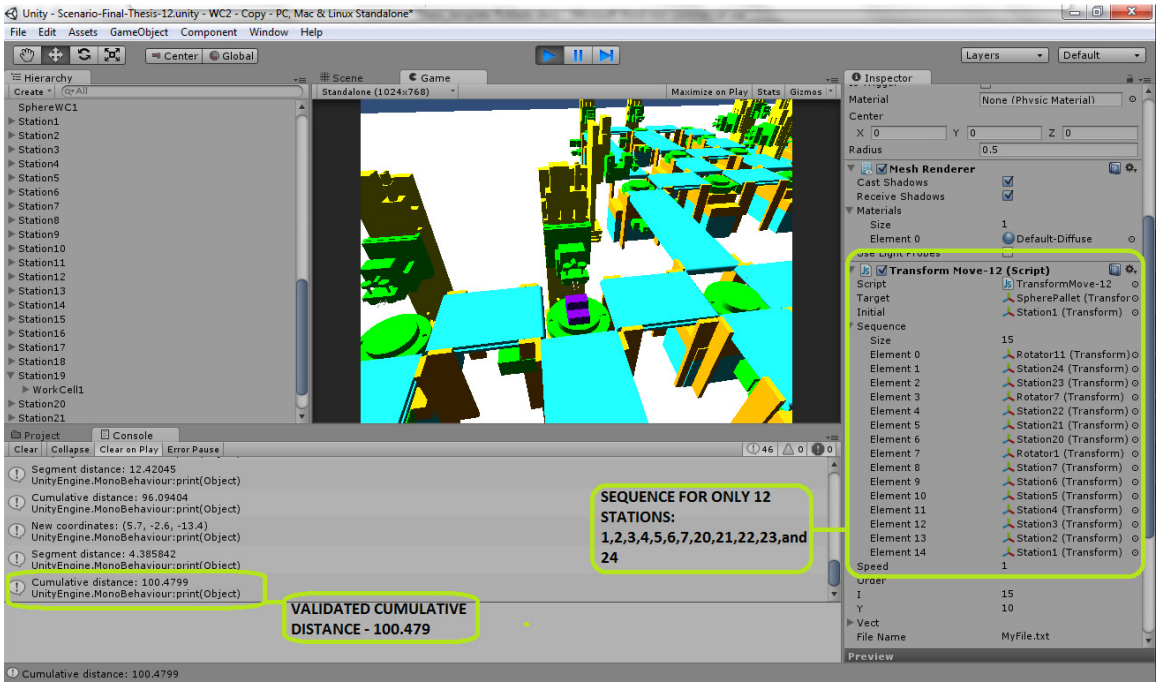

Figure 19. Near optimal sequence for twelve station layout - validated in Unity®

Another scenario is when there are coordinates available for a group of micro assembly stations but they have not been connected by any conveyors. This scenario applies to the early stages of the micro factory design because the engineers could find near optimal distribution paths prior to linking the stations in any way. For example, if we take the coordinates of the 24 stations used in our original model and assume that they are not connected by any conveyors, the distance matrix would be composed of straight paths between all the stations. See Table 22 below.

Table 22. Distance Matrix for layout without conveyors

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	100	4.386	13.17	12.42	13.19	8.753	5.667	9.445	21.27	22.6	21.79	18.76	24.14	26.36	29.56	32.87	36.37	31	26.8	21.01	24.41	20.76	15.09	13.12
2	4.386	100	12.42	13.17	15.24	11.61	9.505	12.14	22.6	24.64	24.71	22.08	28.04	29.97	32.83	36.37	40.05	35.24	31	25.4	28.28	24.41	18.81	17.27
3	13.17	12.42	100	4.386	8.831	10.09	12.54	22.55	34.37	35.75	34.56	31.17	35.16	38.05	41.81	44.65	47.7	40.05	36.37	28.27	25.4	21.01	16.94	18.2
4	12.42	13.17	4.386	100	4.445	6.601	9.953	21.22	33.52	34.37	32.54	28.91	32.13	35.28	39.3	41.85	44.65	36.37	32.87	24.41	21.01	16.63	12.68	14.33
5	13.19	15.24	8.831	4.445	100	4.88	8.905	20.75	33.22	33.51	30.99	27.16	29.42	32.82	37.12	39.3	41.81	32.83	29.56	20.7	16.57	12.18	8.497	10.8
6	8.753	11.61	10.09	6.601	4.88	100	4.025	15.87	28.34	28.68	26.35	22.6	25.54	28.72	32.82	35.28	38.05	29.97	26.36	18.2	17.27	13.12	7.857	8.181
7	5.667	9.505	12.54	9.953	8.905	4.025	100	11.85	24.32	24.71	22.6	18.96	22.62	25.54	29.42	32.13	35.16	28.04	24.14	16.94	18.81	15.09	9.432	7.857
8	9.445	12.14	22.55	21.22	20.75	15.87	11.85	100	12.47	13.22	12.57	10.11	16.95	18.22	20.73	24.43	28.29	25.4	21.01	18.55	26.56	24.07	18.9	15.32
9	21.27	22.6	34.37	33.52	33.22	28.34	24.32	12.47	100	4.386	9.505	11.61	18.81	17.27	16.57	21.01	25.4	28.27	24.41	26.6	37.13	35.39	30.7	26.82
10	22.6	24.64	35.75	34.37	33.51	28.68	24.71	13.22	4.386	100	5.667	8.753	15.09	13.12	12.18	16.63	21.01	24.41	20.76	24.11	35.39	34.13	29.9	25.9
11	21.79	24.71	34.56	32.54	30.99	26.35	22.6	12.57	9.505	5.667	100	4.025	9.431	7.857	8.496	12.68	16.94	18.81	15.09	18.94	30.7	29.9	26.22	22.2
12	18.76	22.08	31.17	28.91	27.16	22.6	18.96	10.11	11.61	8.753	4.025	100	7.857	8.18	10.8	14.32	18.2	17.27	13.12	15.36	26.82	25.9	22.2	18.18
13	24.14	28.04	35.16	32.13	29.42	25.54	22.62	16.95	18.81	15.09	9.431	7.857	100	4.025	8.905	9.953	12.54	9.505	5.667	11.9	24.32	24.71	22.6	18.96
14	26.36	29.97	38.05	35.28	32.82	28.72	25.54	18.22	17.27	13.12	7.857	8.18	4.025	100	4.88	6.601	10.09	11.61	8.753	15.92	28.34	28.68	26.35	22.6
15	29.56	32.83	41.81	39.3	37.12	32.82	29.42	20.73	16.57	12.18	8.496	10.8	8.905	4.88	100	4.445	8.831	15.24	13.19	20.8	33.22	33.51	30.99	27.16
16	32.87	36.37	44.65	41.85	39.3	35.28	32.13	24.43	21.01	16.63	12.68	14.32	9.953	6.601	4.445	100	4.386	13.17	12.42	21.27	33.52	34.37	32.54	28.91
17	36.37	40.05	47.7	44.65	41.81	38.05	35.16	28.29	25.4	21.01	16.94	18.2	12.54	10.09	8.831	4.386	100	12.42	13.17	22.6	34.37	35.75	34.56	31.17
18	31	35.24	40.05	36.37	32.83	29.97	28.04	25.4	28.27	24.41	18.81	17.27	9.505	11.61	15.24	13.17	12.42	100	4.386	12.17	22.6	24.64	24.71	22.08
19	26.8	31	36.37	32.87	29.56	26.36	24.14	21.01	24.41	20.76	15.09	13.12	5.667	8.753	13.19	12.42	13.17	4.386	100	9.485	21.27	22.6	21.79	18.76
20	21.01	25.4	28.27	24.41	20.7	18.2	16.94	18.55	26.6	24.11	18.94	15.36	11.9	15.92	20.8	21.27	22.6	12.17	9.485	100	12.42	13.17	12.54	10.09
21	24.41	28.28	25.4	21.01	16.57	17.27	18.81	26.56	37.13	35.39	30.7	26.82	24.32	28.34	33.22	33.52	34.37	22.6	21.27	12.42	100	4.386	9.505	11.61
22	20.76	24.41	21.01	16.63	12.18	13.12	15.09	24.07	35.39	34.13	29.9	25.9	24.71	28.68	33.51	34.37	35.75	24.64	22.6	13.17	4.386	100	5.667	8.753
23	15.09	18.81	16.94	12.68	8.497	7.857	9.432	18.9	30.7	29.9	26.22	22.2	22.6	26.35	30.99	32.54	34.56	24.71	21.79	12.54	9.505	5.667	100	4.025
24	13.12	17.27	18.2	14.33	10.8	8.181	7.857	15.32	26.82	25.9	22.2	18.18	18.96	22.6	27.16	28.91	31.17	22.08	18.76	10.09	11.61	8.753	4.025	100

FINAL DISTANCE MATRIX - No Conveyors

When this scenario is solved, MatLab yields the following near optimal sequence: 1 2 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 5 4 3 6 7 1. This sequence results in a cumulative distance of 160.869 as displayed in Figure 20 below.

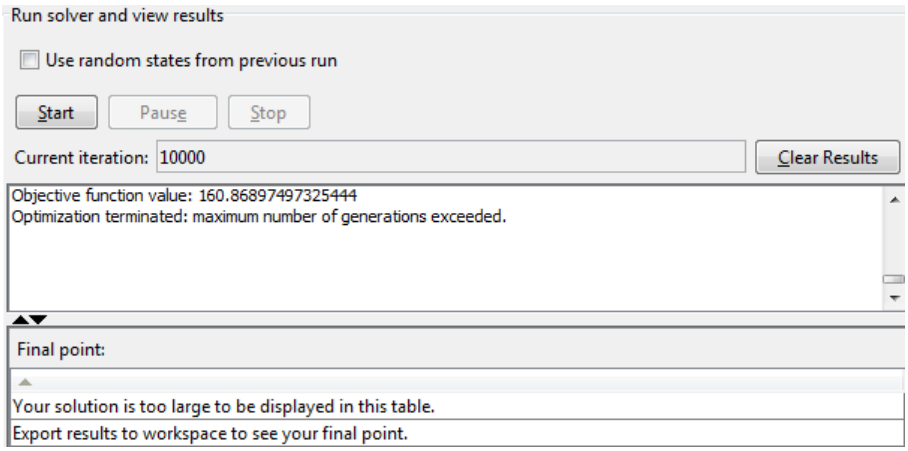

Figure 20. MatLab's near optimal fitness value for the layout without conveyors

This sequence is simulated in the Unity environment and the near optimal cumulative value is also validated as displayed in Figure 21 below. The twenty four station sequence can be seen to the right of the figure while the final cumulative distance value can be seen in the lower left corner. Notice that this value is 160.869 which matches the value obtained in MatLab. This value represents an improvement over the 184.5829 value that is obtained with the layout that connects the same twenty four stations with conveyors.

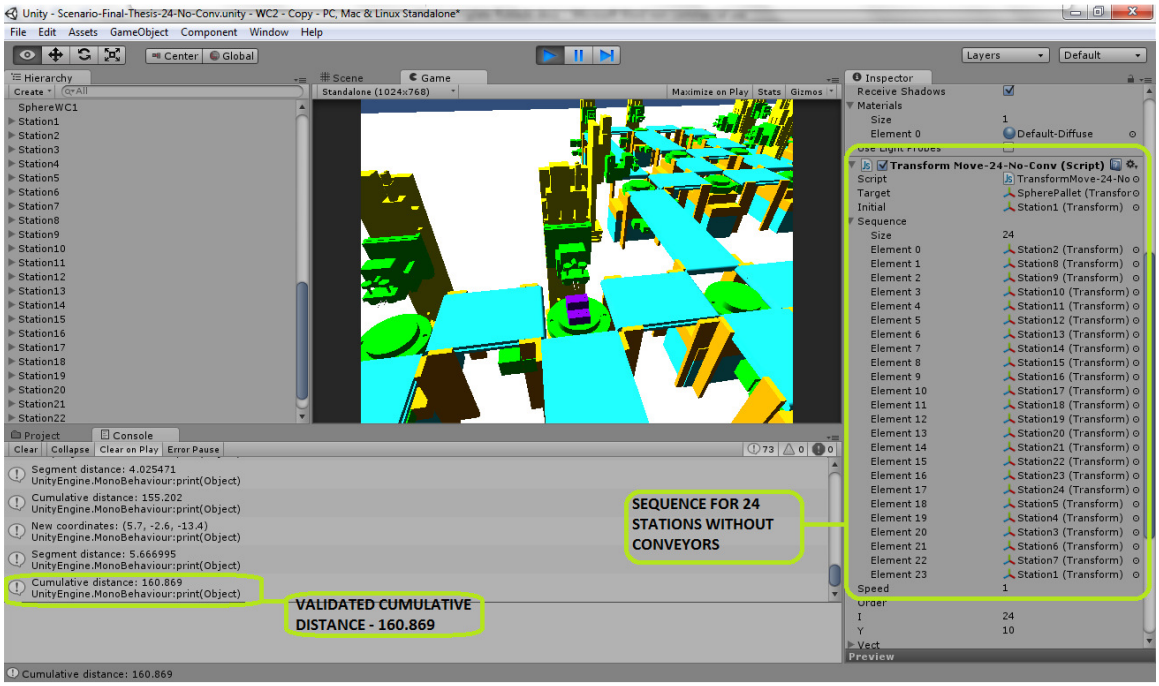

Figure 21. Near optimal sequence for no-conveyor layout - validated in Unity®

This information can be used by a micro assembly factory designer to select a better arrangement of conveyors or other devices to connect the stations. As a matter of fact, the twenty four micro assembly station layout would change significantly reducing the number of conveyors and rotators used to connect them all if the stations are rearranged to be connected following this last near optimal sequence. Figure 22 shows how the layout that yields a cumulative distance of 160.869 would look like. Unity is very flexible to move and reposition objects. This helps to quickly recreate different layout options that derive from the improvement efforts. Notice how different this layout is if compared to the original layout (Figure 14) that is connected by conveyors [47].

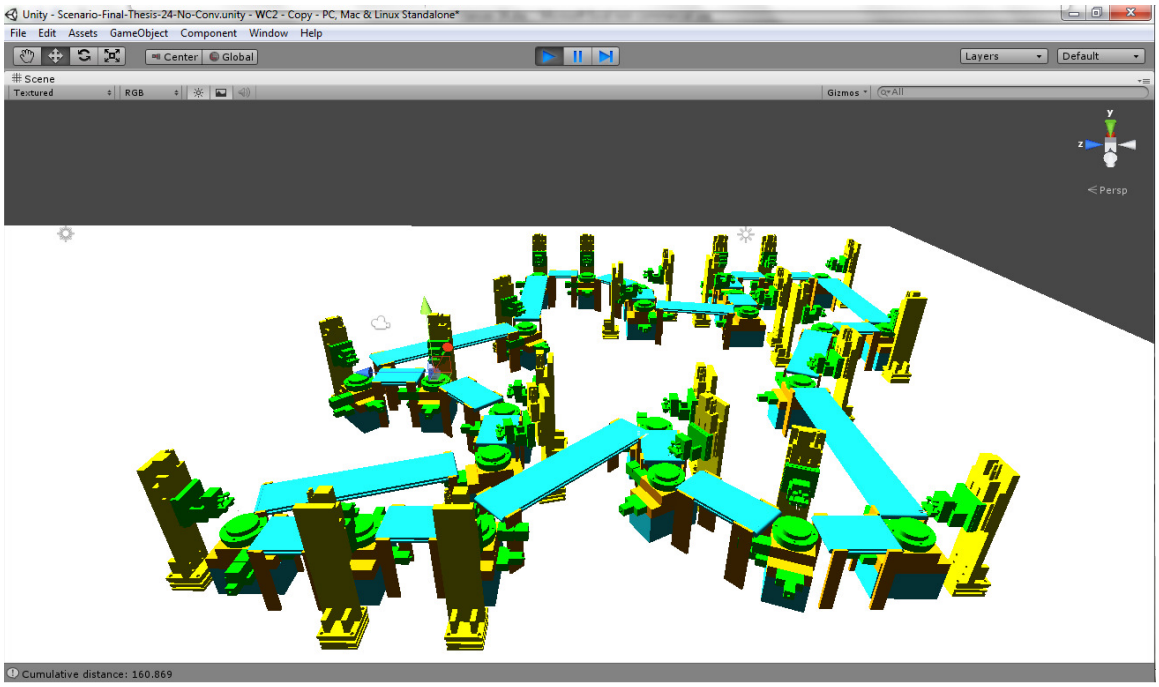

Figure 22. Twenty four stations rearranged after potential improvement analysis

3.9 Methodology Conclusion

The proposed methodology is used successfully to create a micro factory virtual reality environment in Unity ® with 24 work cells. The model is successfully used to simulate several sequences for different circumstances:

- Material distribution for a twenty four cell layout connected by conveyors and rotators.
- Material distribution for only twelve of the twenty four available stations.
- Design of material distribution sequence to supply twenty four work cells that are not limited by conveyor connections.

For all of these scenarios, Excel, MatLab and Unity are used in unison to create distance matrices, find near optimal distribution sequences by applying genetic algorithm techniques and simulate the movement of materials along the near optimal sequences with a virtual reality model.

In all situations the cumulative travel distances calculated in the Unity model matched the objective function value estimated in MatLab. This validated the ability of the model to accurately represent the motion of materials within a micro assembly factory.

Finally, this methodology can be used not only to study and improve existing micro factory systems but to also design future micro factories to be more efficient. The flexibility of the Unity environment enables the users not to only simulate the movement of materials along near optimal sequences but to also reposition objects to quickly create different layout options.

CHAPTER IV

DETAILS OF MODELING IN UNITY

4.1 Introduction

The most important element of this project is the model constructed with Unity ®. The model enables us to simulate different scenarios associated with Micro Assembly factories and to validate near optimal material distribution sequences found with genetic algorithms using MatLab.

This chapter describes the details and techniques used to build the model used for this project. Game object importing, creation, positioning, nesting, and scripting are discussed with the intention to provide a guideline to produce similar models.

4.2 Object Importing from External Sources

Objects used to build a Layout in Unity® can be imported from external sources such as CAD software when they are converted to a '.3ds' format [48]. Not all CAD software can save a model directly with a '.3ds' extension; therefore, conversion software may be needed. A good option for converting models to this extension is to use the 3D-Tool and the 3D-NativeCAD Converter software available at http://www.3d-tool.com/ [49].

In order to display the process to import objects from external sources Figures 23, 24 and 25 below show screenshots from a rod imported from a SolidEdge model [50]. The rod is converted in the 3D-Tool environment and then may be used in Unity as an object to construct a material pin that travels from station to station throughout the micro factory. The model from SolidEdge is opened with 3D-Tool software and then saved as a '3ds' model in the Unity assets folder. Once this is done, the newly converted model is available for use inside Unity. Figure 26 below displays the final use of a rod as a section of a travelling pin [47].

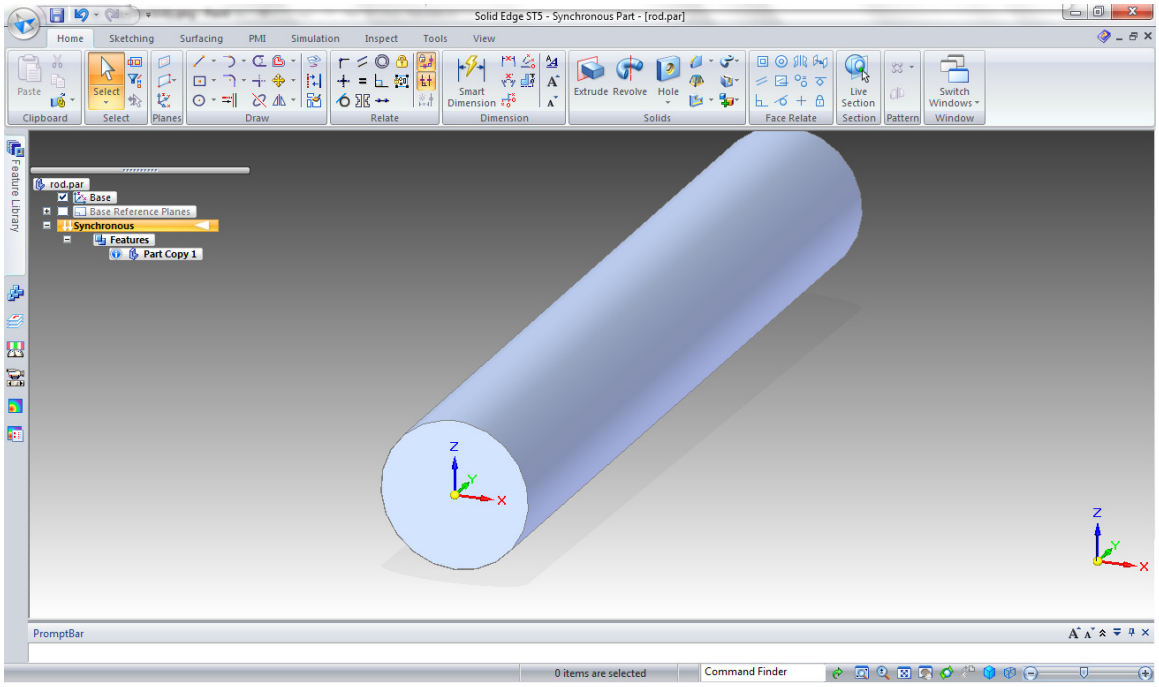

Figure 23. Solid Edge Part to be imported into Unity®

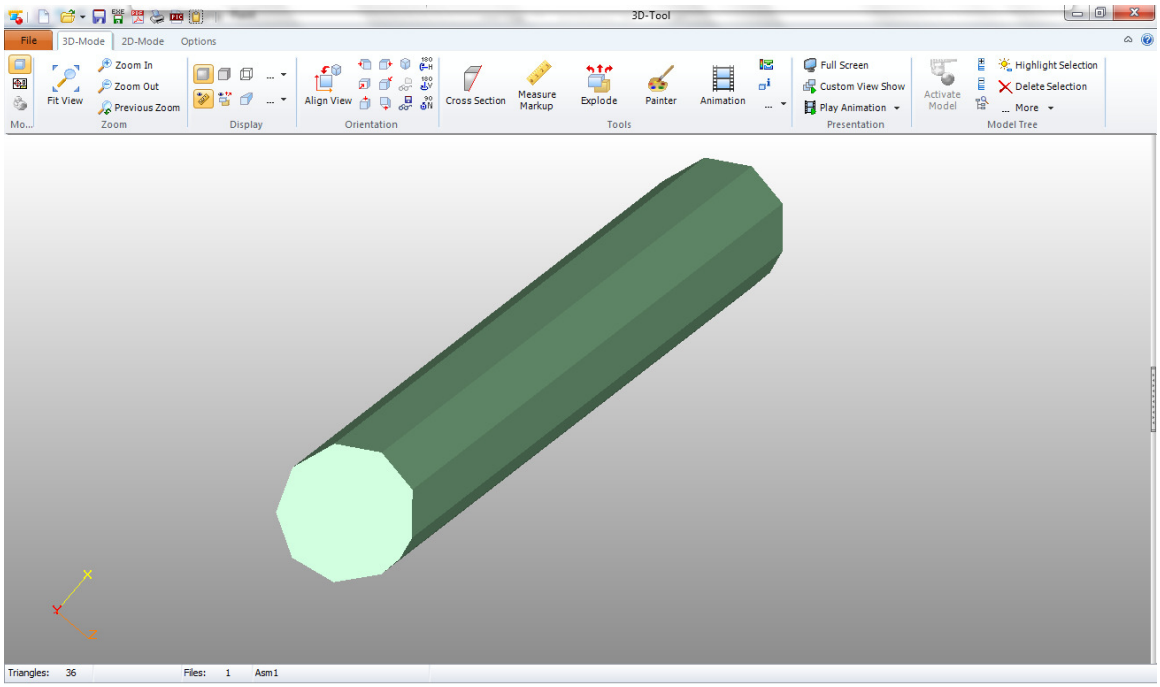

Figure 24. Solid Edge Part opened with 3DTool to be converted to '.3ds' format

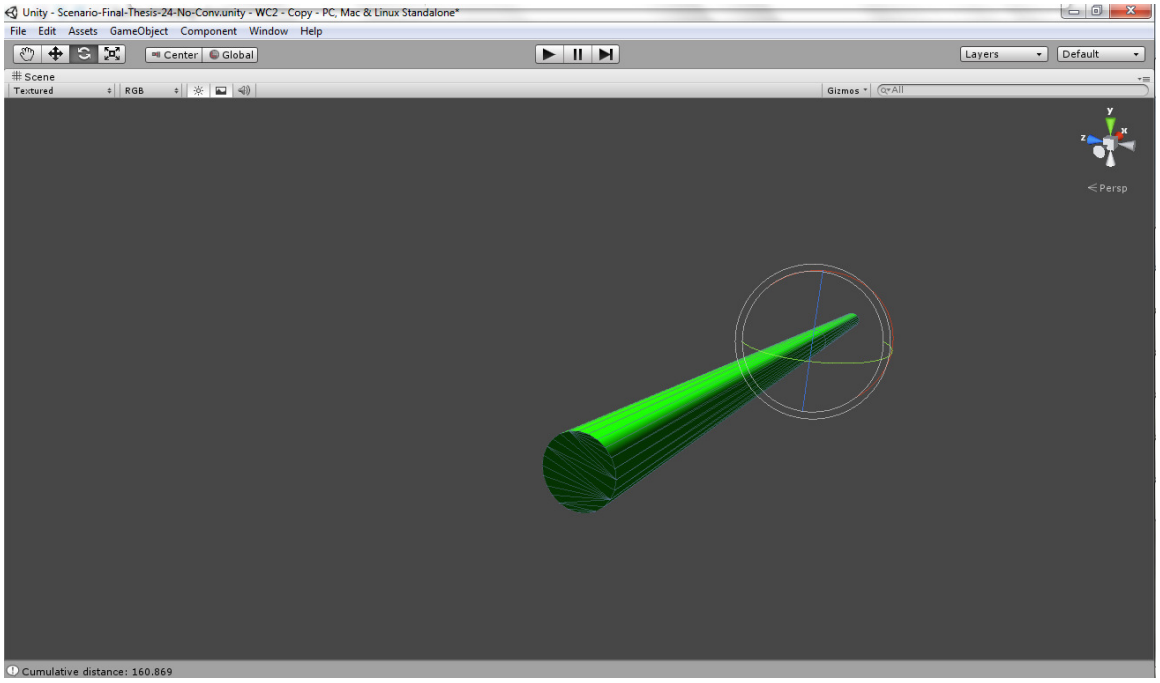

Figure 25. Converted '.3ds' part imported into Unity®

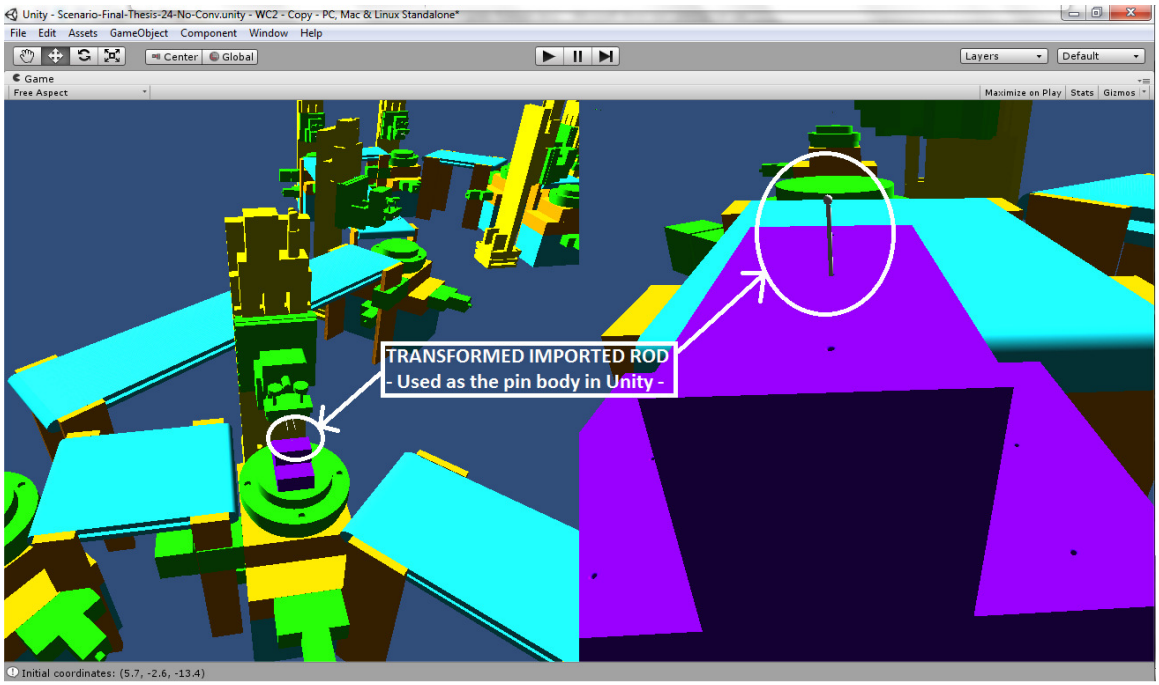

Figure 26. Converted '.3ds' part used in Unity®

4.3 Object Creation with Unity Resources

Unity also has resources to create objects. One of the first routes to create objects is to use the 'Create Other' menu drop down list as displayed in Figure 27 below [51-53]. A cube is created in the Unity Scene view and it can be seen to the right of the drop down menu.

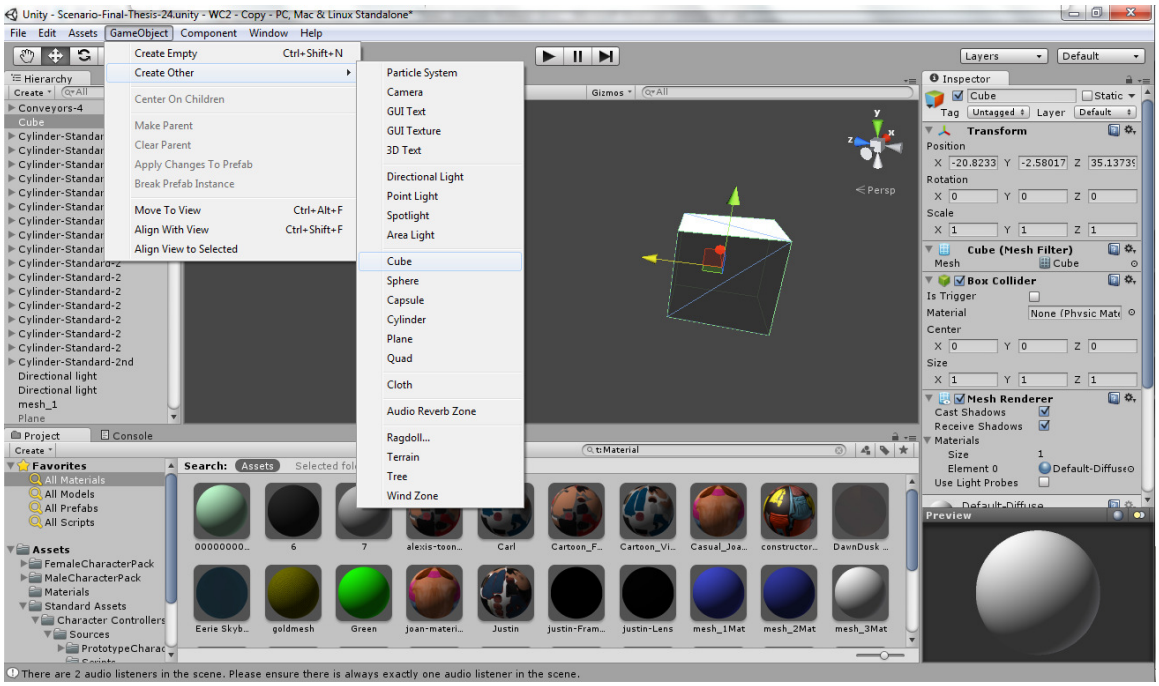

Figure 27. Creating Objects with Unity® resources

Once the object that is needed is created, it can be shaped with the transform options inside the Inspector view or with the transform tools displayed on the upper left corner of the main Unity screen. These tools enable users to change the scale, orientation and the position of the object by either entering different values into the inspector transform fields or by clicking and dragging the translation, rotation and/or scaling gizmos directly in the scene view. Figure 28 below displays two cubes that are identical prior to scaling and rotating one of them. Notice that the X, Y and Z scale values in the Inspector view have been changed to 0.5 each which scales the cube down to 50% of its size in all dimensions. Similarly, the cube is rotated 45 degrees in both, X and Y axis which positions it angled in reference to the first original cube [53].

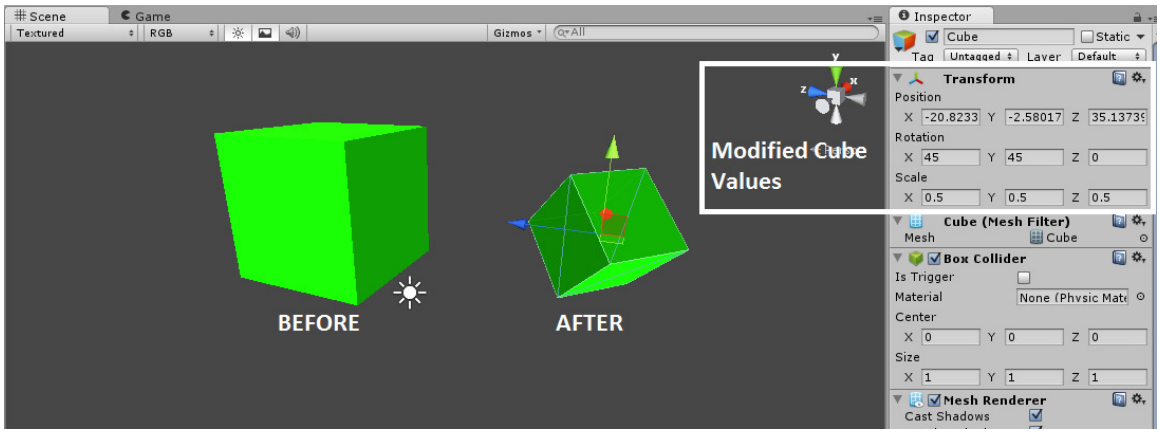

Figure 28. Modifying Objects Inside Unity®

Another way to modify the shape and position of the objects is to use the gizmos and transformation tools as displayed in Figure 29 below. These tools are used directly by clicking and dragging with the mouse. When the axis of interest is clicked on it changes to yellow. Once the axis is yellow it can be dragged to change the characteristics of the object. In Figure 29, from left to right, the effect of scaling is displayed first followed by the effects of rotation and translation [53, 54].

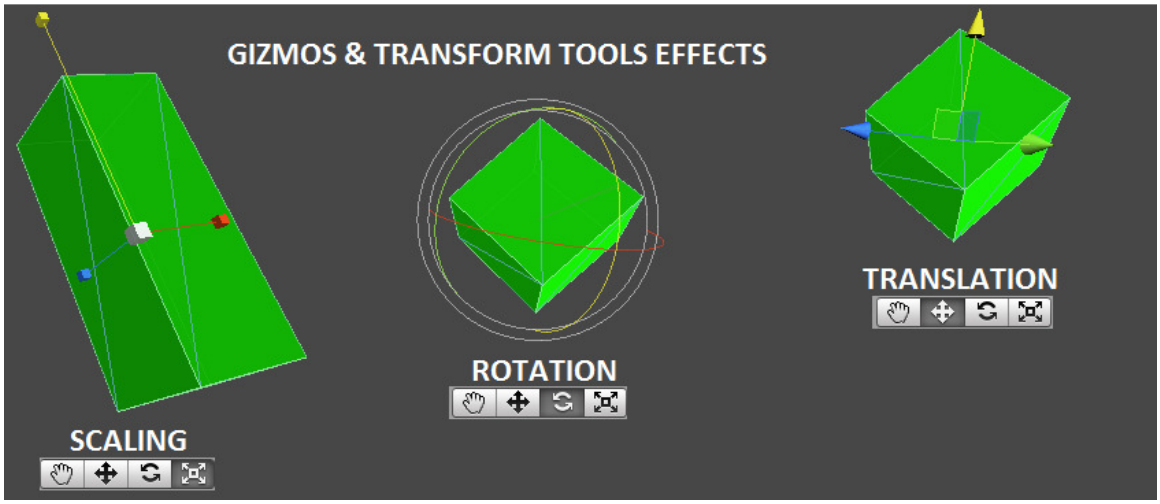

Figure 29. Transforming gizmos and tools inside Unity®

By using these tools, game (layout) objects can be created directly inside Unity®. This is a useful resource especially when used in combination with imported objects that have been created to scale using CAD software. Unity objects can be created to match the scale of the imported items.

Unity also has an additional way to create objects that are pre-fabricated and/or available through a Unity® Asset Store. As displayed in Figure 30, these prefabs can be accessed through the Project View and they may be dragged into the scene. The Unity Asset Store also enables the users to search and download for a variety of assets and prefabs that may be useful for a given project. This method is not used to create objects for this particular project; however, it is worth to mention its availability [55, 56].

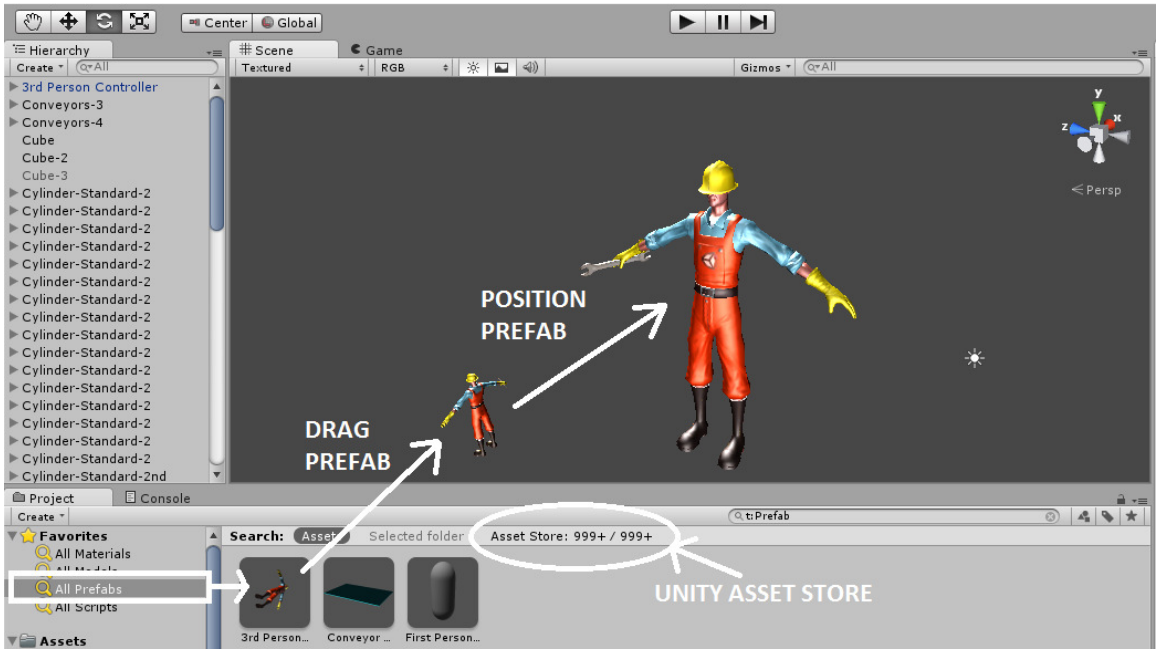

Figure 30. Prefabs and Unity® asset store.

4.4 Positioning

Although the previous section introduced concepts related to the positioning of an object created inside Unity®, it is still important to discuss the coordinate systems that are used to position all objects within the game (layout) and other useful positioning techniques such as vertex snapping.

In Unity® there is a three dimensional coordinate system that is used to place game objects within the space. All objects have a coordinate value for each of the three axes (X, Y, Z) that reference to the object's position within the overall coordinate system. For example, in Figure 31 below, the cube has the following coordinates: X = 10.69041; Y = -3.94778; Z = 11.48382.

Each object also has a local coordinate system that is represented by the, previously discussed, gizmos. Understanding the concept of local coordinates is particularly important when two objects are nested and one becomes the parent of the other. In this situation, the child transformations will be relative to the parent's transformations [53].

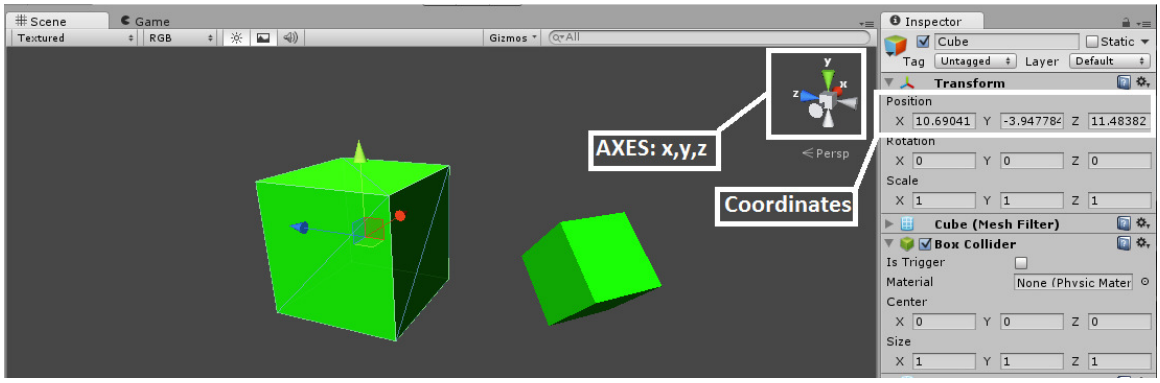

Figure 31. Three dimensional coordinate system

A tool that is particularly important while building the model used for this project is the vertex snapping tool. Vertex snapping enables the user to quickly connect objects by joining their vertices. In order to use vertex snapping, the user must press and hold the "V" key and left-click with the mouse to select the vertex by which the object will be dragged to the desired destination. Once the vertex has been selected, the user must hold the mouse and the "V" key down while moving the object to the target location. When the cursor is over the destination object, each of the destination vertices will be highlighted as the dragged object is brought nearby. The user places the dragged object by releasing the mouse and "V" key as the desired destination vertex is highlighted. Figure 32 below shows this process [57].

Unity also has other snapping capabilities that are not used for this project but that may be useful for other projects. These capabilities include unit snapping, surface snapping, and Look-At Rotation. More details on these capabilities can be found in the references provided in this project [57].

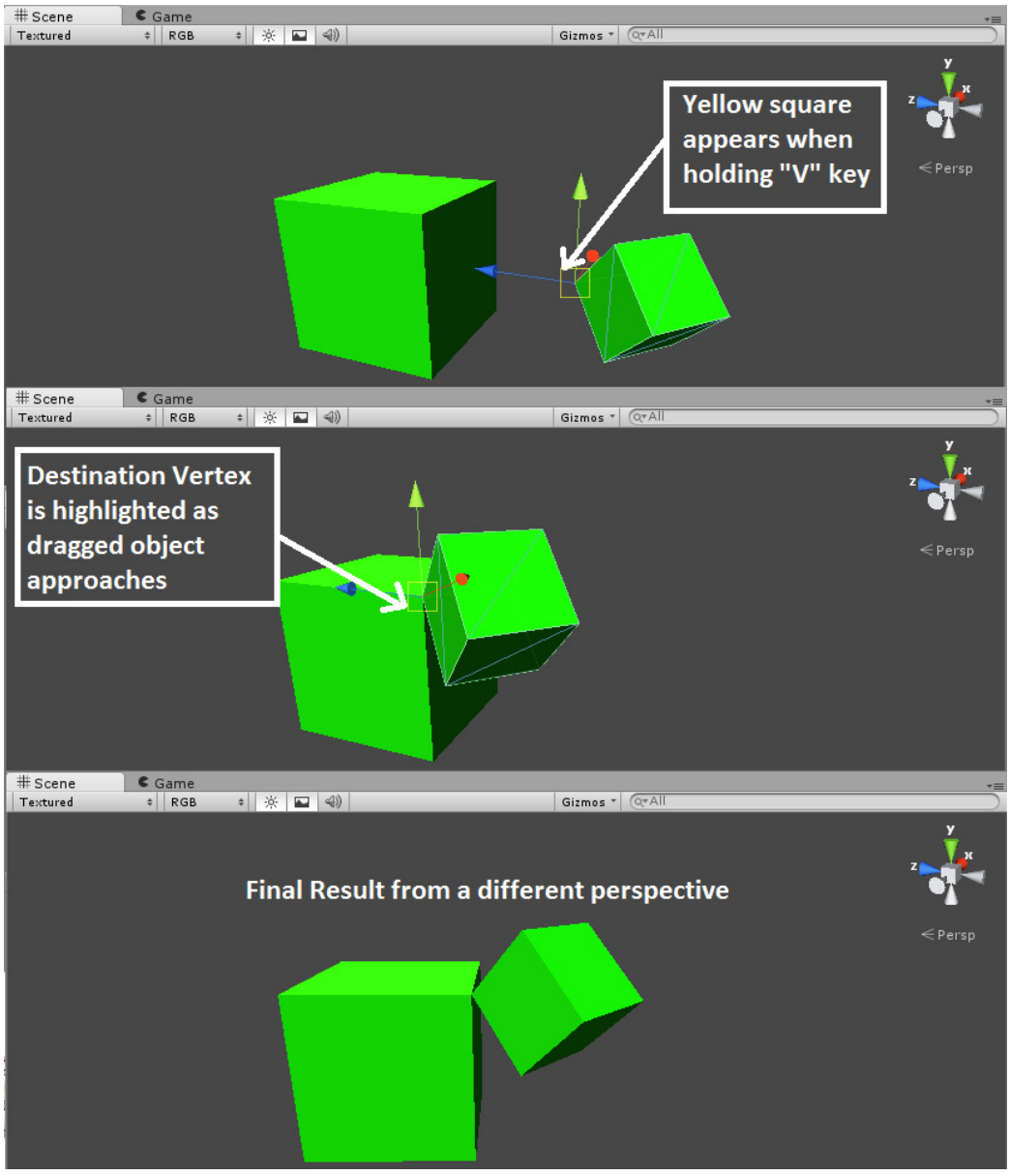

Figure 32. Vertex Snapping Process

4.5 Nesting

The technique used to create the micro assembly stations from several independent objects and make them all behave as one whole item is called nesting [58]. The cubes that we have been using as examples throughout this chapter can help us to understand and see how nesting works. In Figure 33 below, both cubes are independent first; then, cube-2 is nested into the first cube by dragging it in the hierarchy window. Lastly, both cubes are displayed as one object after being nested.

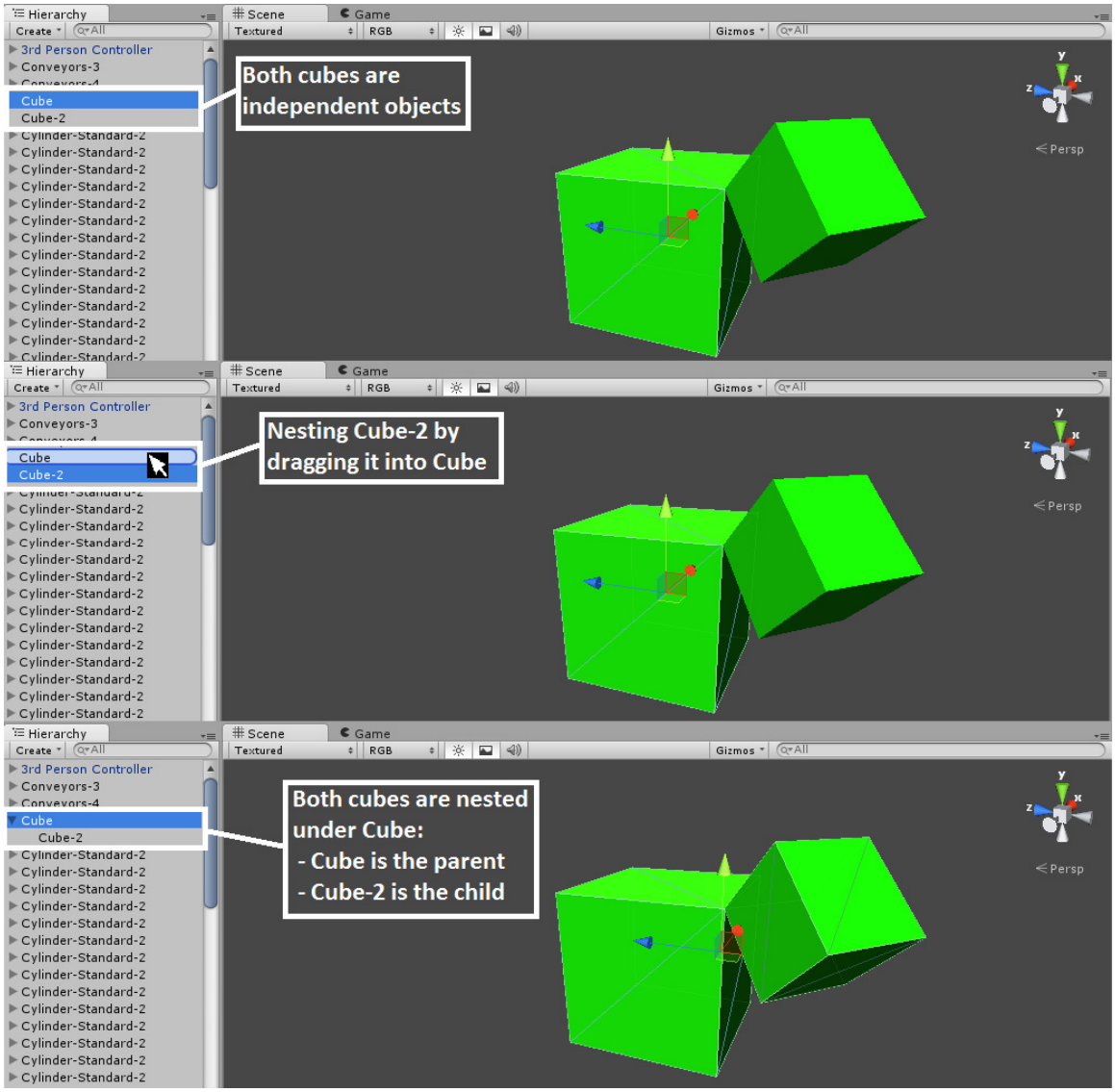

Figure 33. Nesting Process

This technique is particularly important to this project because all the stations have been created and nested under a network of spherical objects that serve as coordinate references for the micro assembly stations. These references act like 'addresses' that enable us to accurately send moving materials (pin pallet) to each of the stations when running the Unity simulation. When objects are nested, one of them becomes the parent while the rest of them become children. In the micro assembly stations the parent object is a small spherical object that is named with the word "Station" and a number. Similarly, the rotators used for this project are nested under a small sphere named with the word "Rotator" and a number.

Figure 34 below displays the station sphere as an independent object before anything is nested under it. As it can be seen, the sphere has been named "Rotator7-Example." Figure 35 displays the same sphere but in reference to the overall space. In this image, it can be appreciated how tiny the sphere really is. These spheres could almost represent any coordinate point within the overall layout and that is precisely why they are used as references to assist the distribution of materials when the simulation is running. Finally, Figure 36 displays the "Rotator7-Example" as a nested whole assembly [47]. All micro – assembly stations in this project are created using the same technique explained here. Next section will explain how simulation scripting takes advantage of this.

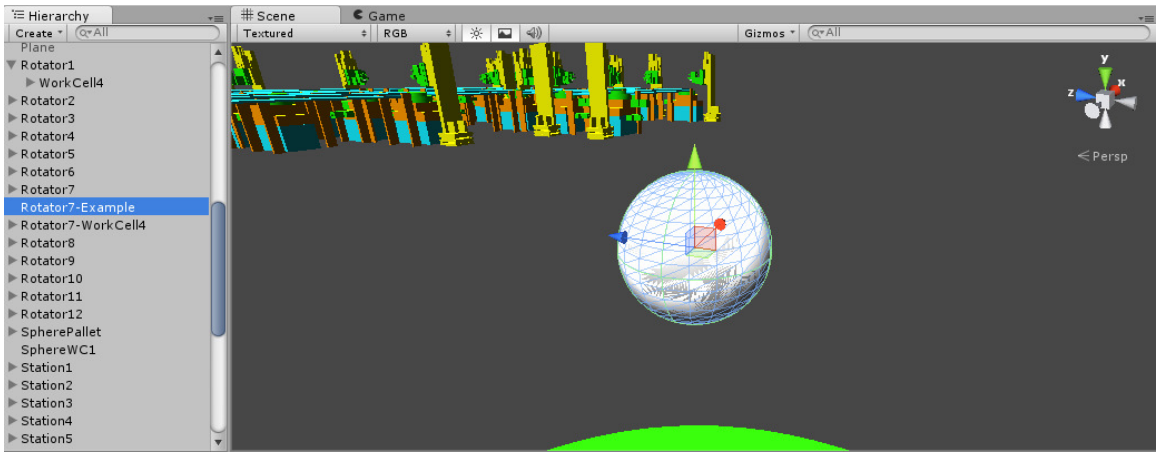

Figure 34. Station sphere used for nesting

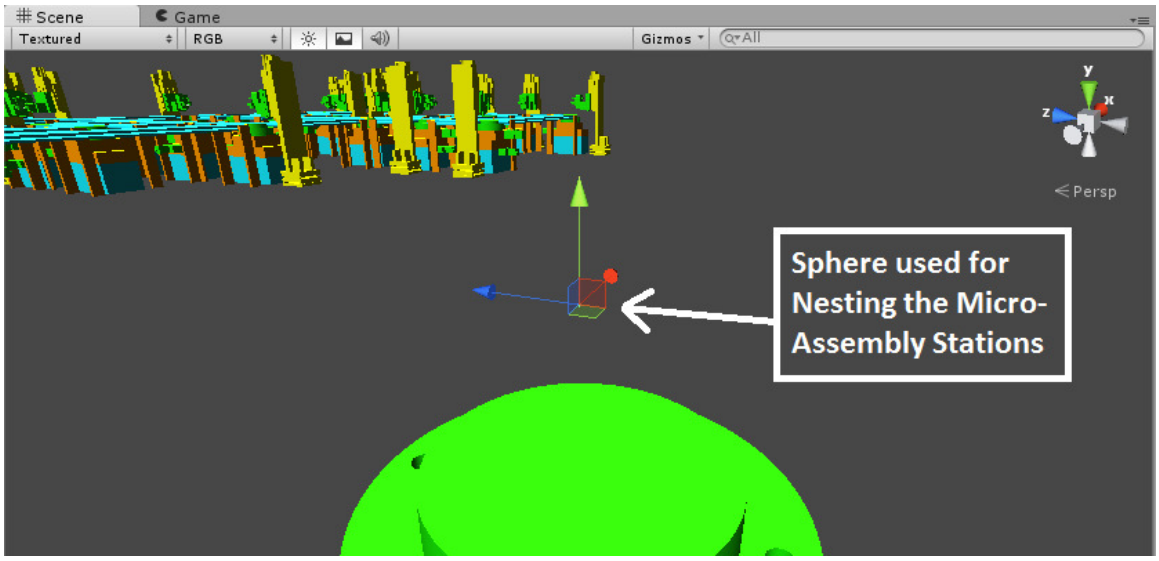

Figure 35. Nesting sphere in reference to the overall space

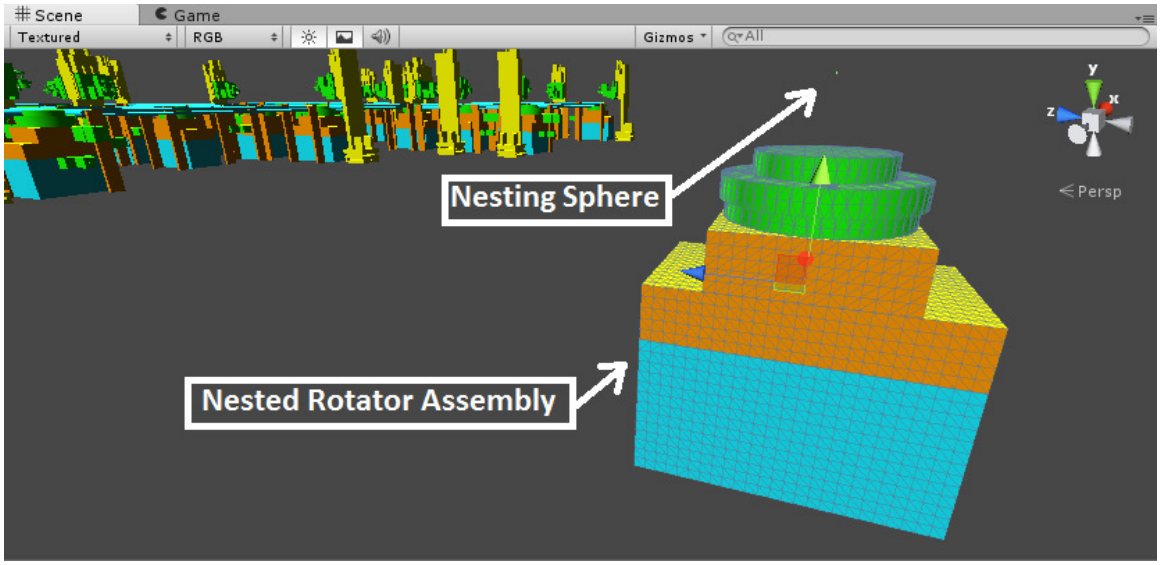

Figure 36. Nested Rotator Assembly and Nesting Sphere

To illustrate the effects that the nesting sphere has over the simulation, the example rotator used in the previous images is placed nearby the actual Rotator7. In Figure 37 a sphere is purposely placed nearby but the station is not nested under it. When the simulation runs, the pin pallet travels directly to the sphere which is in open space. In Figure 38, the sphere is located

appropriately above the micro assembly station and nesting is performed. As it can be seen, the pin pallet travels directly to the nested station in this figure [47].

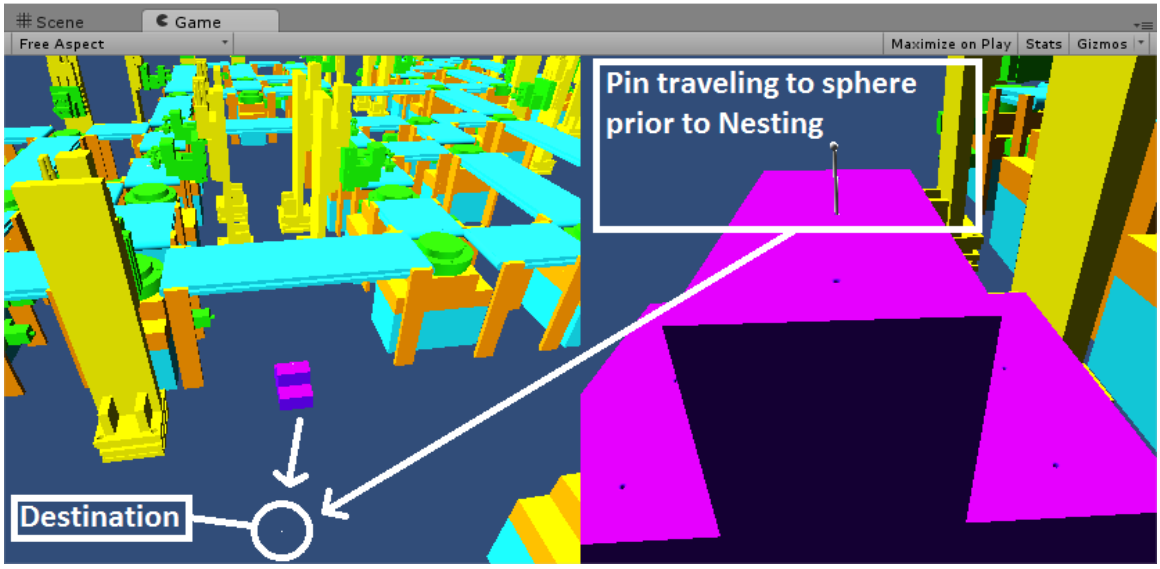

Figure 37. Pin pallet travels to open space towards sphere that is not nested

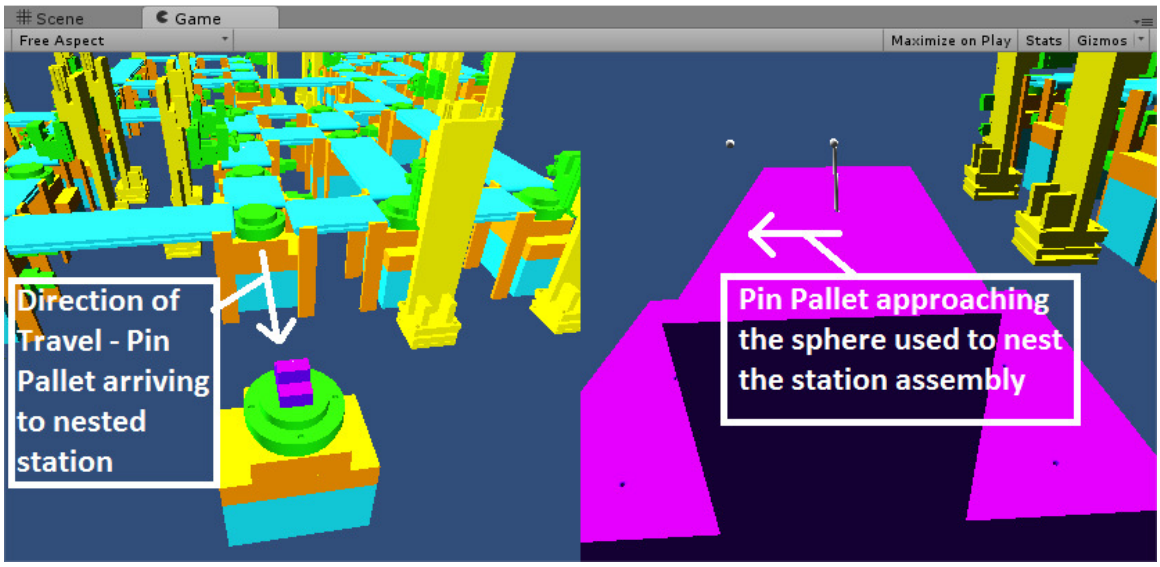

Figure 38. Pin pallet travels to open space towards sphere that has been nested

Next section will explain the scripting that has been used to simulate the motion of materials from one place to another and that is used to create the examples that we have used to explain the important concept of nesting.

4.6 Scripting

This project uses java scripting within Unity® to simulate a controlled sequence of movements of materials (target object) throughout the Micro-Assembly factory.

The scripting code for this project is attached to the Unity® object that represents the pin pallet that moves from station to station in the layout. The code tells the pin pallet where to move next based on a sequence input by the user. The code also gives the user the ability to define the initial station from where the pin pallet will start moving.

The user inputs the information by selecting a transform from the pop up list that opens when clicking on the sequence field or by dragging the object that represents the station to the input fields displayed in the inspector window as seen in Figure 39 below. Once the user input is ready, the script will move the pallet from station to station using a “for” cycle that also calculates the distance traveled by the pin pallet for each segment in the sequence and cumulatively. Such information can be seen in the Console window each time the pin pallet travels one segment from station to station and at the end of the simulation. The distance travelled matches the distance calculated by the algorithm in MatLab where the simulated near optimal sequence is obtained from.

The script used is named “TransformMove” In Unity ® and is displayed in Figure 40 below as it looks when the scripting editor (MonoDevelop) is opened. We can use this figure as a reference as we proceed to explain each code line in the following paragraphs. Script is also provided in the Appendix.

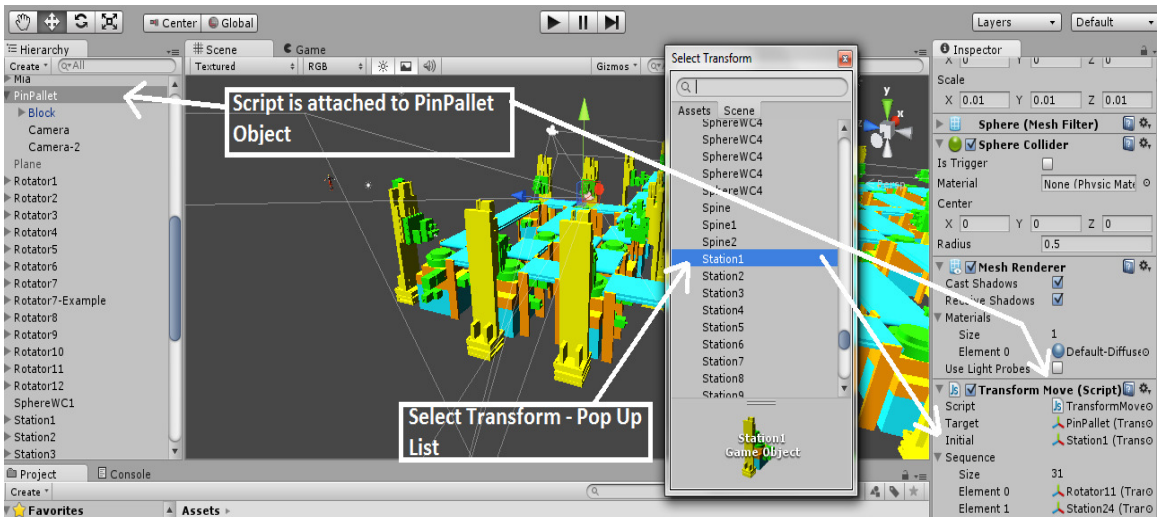

Figure 39. Attachment of Script and Select Transform Pop-Up List

TransformMove.js

TransformMove ▸ FixedUpdate()

```
1  import System;          //Initializing code
2  import System.IO;       //Initializing code
3
4  var target : Transform;  //object to be moved
5  var Initial: Transform;  //object to be the initial place from where the target object starts moving
6  var Sequence: Transform[]= new Transform[31]; // array that represents the sequence of stations where target moves
7  var speed : float; //Variable that defines how fast the target moves
8  var i = 0; //Counter for the "for" cycle that moves the target from one station to another
9  var vect: Vector3; //vector definition
10 private var dist : float; //Variable used to calculate the distance traveled on each sequence segment
11 private var Totdist : float;//Variable used to calculate the overall cummulative distance
12
13 target.position = Initial.position; //Moves target to the initial position
14 print("Initial coordinates: "+Initial.position);//Prints initial position coordinates in the Console window
15
16 for (i=0; i < Sequence.Length; i++)//cycle that moves target until sequence is completed
17  {
18     dist = Vector3.Distance(Sequence[i].position,target.position);//Calculates segment distance traveled by target
19     Totdist=Totdist+dist;//Calculates cummulative traveled distance
20     vect = Sequence[i].position; //Redefines vect as next (i) sequence position
21     yield WaitForSeconds((dist/speed)+1);//Introduces a pause for target to stop at each station in sequence
22     print("New coordinates: "+target.position); //Prints coordinates for each station where the target stops by
23     print("Segment distance: "+dist);      //Prints segment distance value
24     print("Cumulative distance: "+Totdist);//Prints cummulative distance value
25     print(vect);//Prints coordinates of sequence object where target arrives into
26  }
27
28 function FixedUpdate() {//Function that constantly updates the status of the model (and target)
29
30     target.position = //New position of target equals:
31     Vector3(Mathf.MoveTowards(target.position.x,vect.x,speed*Time.deltaTime),//moving in X at speed rate
32     Mathf.MoveTowards(target.position.y,vect.y,speed*Time.deltaTime), //moving in Y at a speed rate
33     Mathf.MoveTowards(target.position.z,vect.z,speed*Time.deltaTime));//moving in Z at a speed rate
34
35 }
```

Figure 40. TransformMove Script as seen in the MonoDevelop editor

The first two lines of code displayed below initialize the script and reference file or directory paths [59].

1. *import System; //Initializing code*
2. *import System.IO; //Initializing code*

The next few code lines define all the classes (variables) used by the Script. There are different types of classes listed. The “target” and “Initial” classes are Transforms which means that they can be objects that represent or do something in the scene [60]. The “Sequence” class is an array of Transforms which represent the arrangement of stations where the target will stop by; its size is defined inside the brackets of the “new Transform[--size here--]” expression [61]. The “speed”, “dist” and “Totdist” variables are float type which means that they are numbers with decimal values when needed. The “i” value is an integer variable that is used as a counter for the “for” cycle that moves the target from station to station. Finally, the “vect” variable is of the Vector3 type which is used as a directional vector that helps the target to move from one location to another with help of the “FixedUpdate” function discussed further below [62].

3. *var target : Transform; //object to be moved*
4. *var Initial: Transform; //object to be the initial place from where the target object starts moving*
5. *var Sequence: Transform[]= new Transform[31]; // array that represents the sequence of stations where target moves*
6. *var speed : float; //Variable that defines how fast the target moves*
7. *var i = 0; //Counter for the "for" cycle that moves the target from one station to another*
8. *var vect: Vector3; //vector definition*
9. *private var dist : float; //Variable used to calculate the distance traveled on each sequence segment*
10. *private var Totdist : float;//Variable used to calculate the overall cumulative distance*

The next lines initialize the position of the target at the position of the initial station and print the initial coordinates in the console window.

11. *target.position = Initial.position; //Moves target to the initial position*
12. *print("Initial coordinates: "+Initial.position);//Prints initial position coordinates in the Console window*

The next few lines run the “for” cycle that moves the target to all the stations listed in the sequence. The “for” cycle increases the value of the variable “i” until the length of the “Sequence” array has been covered. Both variables, “dist” and “Totdist” are calculated and updated for each “i” value; the “dist” variable is calculated by using the “Vector3.Distance” function [62] while the “Totdist” variable is updated by cumulatively adding “dist.” The “yield

WaitForSeconds((dist/speed)+1)" introduces a pause for the target to stop at each station that is visited in the sequence; this way the pin pallet arrives, stops, then it continues its path [63]. Finally, the "for" cycle prints information in the console window including the new coordinates that the target obtains at each step of the sequence, the segment distance, the cumulative distance and the coordinates of the "vect" variable which represent the last vector used in the cycle.

```
18. for (i=0; i < Sequence.Length; i++)//cycle that moves target until sequence is completed
19. {
20. dist = Vector3.Distance(Sequence[i].position,target.position);//Calculates segment
    distance traveled by target
21. Totdist=Totdist+dist;//Calculates cumulative traveled distance
22. vect = Sequence[i].position; //Redefines vect as next (i) sequence position
23. yield WaitForSeconds((dist/speed)+1);//Introduces a pause for target to stop at each
    station in sequence
24. print("New coordinates: "+target.position); //Prints coordinates for each station where
    the target stops by
25. print("Segment distance: "+dist);      //Prints segment distance value
26. print("Cumulative distance: "+Totdist);//Prints cumulative distance value
27. print(vect);//Prints coordinates of sequence object where target arrives into
28. }
```

Lastly, the following lines conform the Update function [64] which is constantly (every frame) updating the condition of the elements in the simulated scenario. It can be seen that the new position of target (target.position) equals the moving X, Y and Z values of the vector (vect) that change thanks to the "Mathf.MoveTowards" function [65] at a speed ratio affected by "Time.deltaTime" [66] in the Vector3 class.

```
29. function FixedUpdate() {//Function that constantly updates the status of the model (and
    target)
30. target.position = //New position of target equals:
31. Vector3(Mathf.MoveTowards(target.position.x,vect.x,speed*Time.deltaTime),//moving in
    X at speed rate
32. Mathf.MoveTowards(target.position.y,vect.y,speed*Time.deltaTime), //moving in Y at a
    speed rate
33. Mathf.MoveTowards(target.position.z,vect.z,speed*Time.deltaTime));//moving in Z at a
    speed rate
34. }
```

4.7 Unity Modeling Conclusion

This chapter helped us to further understand how environments and objects in Unity can be built with resources provided by the game engine or can be exported from files created with CAD software that are converted to a '.3ds' extension. We discussed how positioning and nesting can be used to modify and group objects in the simulated scene. We also have seen how, once the micro assembly cell models are created or imported into Unity, the simulation part is accomplished with scripting. The behavior of objects and responses to user inputs are possible thanks to the scripts that are attached to objects within the virtual environment [29].

CHAPTER V

RESULTS SUMMARY AND CONCLUSION

5.1 Introduction

This project has given us a very powerful tool to simulate Micro Assembly Factories and study the improvement of material flow within them in a virtual and computerized environment. The methodology proposed in this project yielded several results that are summarized in the next section in an effort to consolidate important observations and assist with the generation of conclusions.

In chapter 1, creating a virtual factory environment with Unity® and using it to explore ways to improve micro assembly processes are introduced as the two main components of the main objective for this project. It is further explained that the model created for this thesis would be used along with genetic algorithms to study and improve the flow of materials between micro assembly cells with the intention to improve overall layout design and/or performance.

Chapter 2 included a literature review of work that has been done in micro assembly and virtual reality domains. Information related to micro assembly papers is presented first followed by information related to virtual reality papers. Finally, a summary of papers related to the application of virtual reality in the field of micro assembly is provided.

Chapter 3 introduced the proposed methodology to fulfill the objectives of this thesis where Excel, MatLab and Unity® are used in unison to create distance matrices, find near optimal distribution sequences using genetic algorithm techniques, and simulate the movement of materials along the near optimal sequences with a virtual reality model. As explained further below in this chapter, the proposed methodology is used successfully to fulfill the project objectives while all the fitness values generated by MatLab for near optimal sequences are validated with the simulation in the Unity® environment.

Chapter 4 highlighted how the most important element of this project is the model constructed with Unity ® because it is capable to simulate different scenarios associated with Micro Assembly factories and to validate near optimal material distribution sequences found with genetic algorithms using MatLab. Chapter 4 described the details and techniques used to build the model in Unity® and concepts like game object importing, creation, positioning, nesting, and scripting are discussed with the intention to provide a guideline to produce similar models in the future.

Finally, this chapter focuses on providing a results summary, future work opportunities and overall project conclusions.

5.2 Results summary

The proposed methodology is used successfully to create a micro factory virtual reality environment in Unity ®. The model is successfully used to simulate several sequences for different scenarios:

- Material distribution for a twenty four cell layout connected by conveyors and rotators.
- Material distribution for only twelve of the twenty four available stations.

- Design of material distribution sequence to supply twenty four work cells that are not limited by conveyor connections.

Table 23 below shows a summary of the results associated to these scenarios. Results obtained by MatLab algorithms and the cumulative value calculated by the simulation in Unity® are displayed next to each other. The near optimal sequence that is obtained by the genetic algorithm and simulated in Unity is provided as well.

It is important to understand that different near optimal sequences could be obtained and simulated; however, the important observation here is that the model constructed in Unity® using the methodology proposed by this project is replicating the values estimated by the MatLab algorithms. This validation is very important because it means that the Unity ® environment can be used to design new or improve existing layouts with high level of confidence.

Table 23. Simulation Results Summary

Scenario	**MatLab Result**	**Unity-Simulated Result**	**Near Optimal Sequence**
Twenty four stations WITH conveyors & rotators	184.5829	184.5829	1, 24, 23, 22, 21, 20, 13, 14, 15, 16, 17, 18, 19, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
Twelve station (only) from the twenty four available	100.47989	100.4799	1, 24, 23, 22, 21, 20, 7, 6, 5, 4, 3, 2, 1
Twenty four stations WITHOUT conveyors & rotators	160.8690	160.8690	1, 2, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 5, 4, 3, 6, 7, 1

5.3 Results discussion

The scenarios summarized in Table 23 cover a variety of circumstances that can be recreated in the virtual environment to assist the user with different goals:

- The first model with twenty four stations could be an existing layout with a given number of work stations and conveyors. The objective could be to find alternatives to re layout the stations or simply to find the best route to distribute materials within it.
- The second scenario deals with the partial utilization of a given group of work stations and the search for the near optimal sequence route to supply the active stations with material.
- The third listed model assumes that the design of a work station layout is started from scratch but a set of coordinates is given for the work stations. This scenario can be used by the designer to test different layouts and it could even be used to propose moving work stations to different locations if needed.

All these scenarios are proof of the flexibility that Unity® provides to the user to experiment with different options as layouts are designed or continuously improved. As it can be seen, the simulated results matched the MatLab results each and every time.

5.4 Importance of virtual environments and prototypes

Nowadays, virtual reality environments are getting importance and its use is starting to get popularity as the future of manufacturing. This popularity comes from the broad collection of advantages that using a virtual prototype has for manufacturing companies and designers during the earliest stages of new product development and process improvement.

Take for example the Ford Motor Company where the technology started to be used a few years back to design production lines more ergonomically and where virtual, immersive models have

proven to add value by reducing quality, assembly and other performance issues during the earliest stages of manufacturing processes and cars design. At Ford, virtual assembly lines and products are constructed and tested before any assembly line is put together to launch a car. This approach enables engineers to solve most of the potential problems that the new production line could have during the earliest stages of the project. Overtime, this fact not only eliminates ergonomic, quality and assembly issues but saves significant amount of resources that, otherwise, would be spent in fixing the issues after they are in place [67-74].

The virtual production line and car prototypes help Ford to experiment earlier and be proactive cross functionally. Virtual prototypes help the company to meet cross functional goals and improve different metrics at a lower cost since changes that are needed can be done quickly in the computerized environment rather than in a physical prototype where rework and additional materials would be likely needed. The advantages obtained from the use of virtual prototypes benefit several functional metrics including quality, productivity, safety and ergonomics [67-74].

5.5 Future work and research opportunities

This project presented an approach to build virtual prototypes in Unity® and use them to support continuous process improvement or better process design. The capabilities that this technology has are still far away from being fully explored. The methodology presented here is just one step forward into the future of manufacturing process and new product design. Future work and research opportunities related to this project include:

- The development of a methodology to take models created by this methodology into a semi-immersive environment.
- Taking advantage of the flexibility and software availability that Unity® has to better connect cross functional teams that are involved in design projects even remotely. Since

Unity® is a gaming engine we could almost say that the engineers in a given team could "play" their project's video game on-line, in the cloud, or any other communication enabling technology that may evolve in the future.

- Expanding this methodology to macro assembly environments where it could be used to study the effect that different assembly processes and product designs have on performance. Quality, productivity, safety and ergonomic factors could be explored within several virtual scenarios that can be easily tweaked within the gaming engine environment that Unity® makes available.

Using virtual reality approaches and technology will continue to gain importance in the design world. Working on projects that explore the utilization of approaches similar to the one proposed by this thesis will

5.6 Overall conclusion

The main objective for this project is to create a virtual factory environment and use it to improve micro assembly processes. The following objectives derived from the main one:

- Create a virtual factory environment using computer graphics (UNITY® software).
- Explore the potential use of genetic algorithms and other methods to improve the routing between work cells.

All of these objectives have been successfully fulfilled by this project where a virtual micro assembly factory is created in Unity to simulate different scenarios that could be improved with the use of MatLab methods. For all of these scenarios, Excel, MatLab and Unity are used in unison to create distance matrices, find near optimal distribution sequences applying genetic algorithm techniques and simulate the movement of materials along the near optimal sequences with a virtual reality model.

In all situations the cumulative travel distances calculated in the Unity model matched the objective function value estimated by the MatLab algorithms. This validated the ability of the model to accurately represent the motion of materials within a micro assembly factory.

Finally, this methodology can be used not only to study and improve existing micro factory systems but to also design future micro factories to be more efficient. The flexibility of the Unity environment enables the users not to only simulate the movement of materials along near optimal sequences but to also reposition objects to quickly create different layout options.

REFERENCES

[1] Cecil, J., 2010. Virtual Engineering. *Momentum Press,* New York, USA.

[2] Shet, S., Revero, R.D., Booty, M. R., Fiory, A. T., Lepselter, M. P., and Ravindra, N.M., 2006. "Microassembly Techniques: A Review." Materials Science and Technology, 1, pp. 451-470.

[3] Cecil, J. J., Vasquez, D. D., and Powell, D. D., 2005. "A review of gripping and manipulation techniques for micro-assembly applications." International Journal Of Production Research, 43(4), pp. 819-828.

[4] Ohba, K., Ortega, J.C., Tanie, K., Rin, G., Dangi, R., Takei, Y., Kaneko, T., and Kawahara, N., 2001. "Micro-observation Technique for Tele-micro-operation." Advanced Robotics, 15(8), pp. 781-798.

[5] Beltrami, I., Joseph, C., Clavel, R., Bacher, J., and Bottinelli, S., 2004. "Micro- and Nanoelectric-discharge Machining." Journal of Materials Processing Technology, 149(1–3), pp. 263-265.

[6] Qin, Y., 2006. "Micro-Forming and Miniature Manufacturing Systems — Development Needs and Perspectives." Journal of Materials Processing Technology, 177(1–3), pp. 8-18.

[7] Wang, J., Tao, X., and Cho, H., 2008. "Microassembly of micro peg and hole using an optimal visual proportional differential controller." Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 222(9), pp. 1171-1180.

[8] Feddema, J. T., Xavier, P., and Brown, R., 2001. "Micro-assembly Planning With Van der Waals Force." Journal Of Micromechatronics, 1(2), pp. 139-153.

[9] Hollis, R. L., and Rizzi, A. A., 2004. "Agile Assembly Architecture: A Platform Technology for Microassembly." Advances, 2004.

[10] Hollis, R., and Gowdy, J., 1998. "Miniature Factories for Precision Assembly." International Workshop on Microfactories. pp.9-14. The Robotics Institute, Carnegie Mellon University.

[11] Gaugel, T., Bengel, M., and Malthan, D., 2004. "Building a Mini-assembly System from a Technology Construction Kit." Assembly Automation, 24(1), pp. 43–48.

[12] Ferreira, A., 2000. "Design of a Flexible Conveyer Microrobot with Electromagnetic Field-based Friction Drive Control for Microfactory Stations." Journal Of Micromechatronics, 1(1), pp. 49-66.

[13] Li, J., Li, Z., and Chen, J., 2007. "An Omni-directional Mobile Millimeter-sized Microrobot with 3-mm Electromagnetic Micromotors for a Micro-factory." Advanced Robotics, 21(12), pp. 1369-1391.

[14] Subramaniyam, M., Park, S., & Park, J., 2010. "Analysis of five- and six-machine micro-factory layouts for micro-pump productivity improvement." Journal of Mechanical Science and Technology, 24(11), pp. 2269-2273.

[15] Shuang, B., Chen, J., and Li, Z., 2008. "Microrobot based micro-assembly sequence planning with hybrid ant colony algorithm." The International Journal of Advanced Manufacturing Technology, 38(11), pp. 1227-1235.

[16] Cecil, J., and Kanchanapiboon, A., 2006. "Virtual engineering approaches in product and process design." The International Journal of Advanced Manufacturing Technology, 31(9), pp. 846-856.

[17] Jayaram, S., Connacher, H., and Lyons, K., 1997. "Virtual assembly using virtual reality techniques." Computer-Aided Design, 29(8), pp. 575-584.

[18] Gomes de Sá, A., and Zachmann, G., 1999. "Virtual reality as a tool for verification of assembly and maintenance processes." Computers & Graphics, 23(3), pp. 389-403.

[19] Wang, Q., & Li, J., 2006. "Interactive visualization of complex dynamic virtual environments for industrial assemblies." Computers in Industry, 57(4), pp. 366-377.

[20] Weidlich, D., Cser, L., Polzin, T., Cristiano, D., and Zickner, H., 2007. "Virtual Reality Approaches for Immersive Design." CIRP Annals - Manufacturing Technology, 56(1), pp. 139-142.

[21] Lin, M., Fu, L., and Shih, T., 1999. "Virtual Factory- A Novel Testbed for an Advanced Flexible Manufacturing System." International Conference on Robotics & Automation. 1999.

[22] Wenbin, Z., Juanqi, Y., Dengzhe, M., Ye, J., and Xiumin, F., 2006. "Production engineering-oriented virtual factory: a planning cell-based approach to manufacturing systems design." The International Journal of Advanced Manufacturing Technology. 28(9), pp. 957-965.

[23] Renard, Y., Lotte, F., Gibert, G., Congedo, M., Maby, E., Delannoy, V., and Lécuyer, A., 2010. "OpenViBE: An Open-Source Software Platform to Design, Test, and Use Brain-Computer Interfaces in Real and Virtual Environments." Presence: Teleoperators & Virtual Environments, 19(1), pp. 35-53.

[24] Gobinath, N., Cecil, J., and Powell, D., 2007. "Micro devices assembly using virtual environments." Journal of Intelligent Manufacturing, 18(3), pp. 361-369.

[25] Ferreira, A., Cassier, C., and Hirai, S., 2004. "Automatic microassembly system assisted by vision servoing and virtual reality." IEEE/ASME Transactions on Mechatronics, 9(2), pp. 321-333.

[26] Cecil, J., and Gobinath, N., 2005. "Development of a virtual and physical work cell to assemble micro-devices", Robotics and Computer-Integrated Manufacturing, 21(4–5).

[27] Ferreira, A., Fontaine, J., and Hirai, S., 2002. "Automation of a Teleported Microassembly Desktop Station Supervised by Virtual Reality." Transactions on Control, Automation, and Systems Engineering. 4(1), pp. 23-30.

[28] Cecil, J., Huber, J., Gobinath, N., and Jacques, J., 2011. "A Virtual Factory Environment to support Process Design in Micro Assembly Domains." Computer-Aided Design & Applications, 8(1), pp. 119-127.

[29] Unity Technologies, 2015, "Creating and Using Scripts", from http://docs.unity3d.com/Documentation/Manual/CreatingAndUsingScripts.html.

[30] The MathWorks Inc., 2012, "Global Optimization Toolbox User's Guide", from http://www.mathworks.com/help/releases/R2012a/pdf_doc/gads/gads_tb.pdf.

[31] The MathWorks Inc., 2015, "Global Optimization Toolbox" from http://www.mathworks.com/help/gads/index.html?searchHighlight=Global%20Optimization%20Toolbox.

[32] Wikipedia, 2015, "Euclidean distance" from http://en.wikipedia.org/wiki/Euclidean_distance.

[33] Massey University, 2012, "Pythagoras Theorem", from http://mathsfirst.massey.ac.nz/Algebra/PythagorasTheorem/pythapp.htm.

[34] Microsoft Office Support, 2015, "SQRT Function", from https://support.office.com/en-NZ/article/SQRT-function-654975c2-05c4-4831-9a24-2c65e4040fdf.

[35] Furey, Edward., 2015, "3D Distance Calculator", from http://www.calculatorsoup.com/calculators/geometry-solids/distance-two-points.php.

[36] Microsoft Office Support, 2015, "INDEX Function", from https://support.office.com/en-NZ/article/INDEX-function-0ee99cef-a811-4762-8cfb-a222dd31368a.

[37] Microsoft Office Support, 2015, "IF Function", from https://support.office.com/en-NZ/article/if-function-0f1a0f81-f6e1-46e4-9cd4-9e65445fc3ab.

[38] Microsoft Office Support, 2015, "SUM Function", from https://support.office.com/en-NZ/article/sum-function-a1fe8ecc-2c5a-4d5a-8db0-80b9f7fcbd6b.

[39] Microsoft Office Support, 2015, "AND Function", from https://support.office.com/en-NZ/article/and-function-c192b0d8-0eeb-4768-a674-f84e6df9aefb.

[40] Microsoft Office Support, 2015, "OFFSET Function", from https://support.office.com/en-NZ/article/offset-function-c8de19ae-dd79-4b9b-a14e-b4d906d11b66.

[41] Microsoft Office Support, 2015, "INDIRECT Function", from https://support.office.com/en-nz/article/INDIRECT-function-21f8bcfc-b174-4a50-9dc6-4dfb5b3361cd?ui=en-US&rs=en-NZ&ad=NZ.

[42] The MathWorks Inc., 2006, "xlsread", from http://www.mathworks.com/help/matlab/ref/xlsread.html?searchHighlight=xlsread.

[43] The MathWorks Inc., 2015, "What is the Genetic Algorithm?", from http://www.mathworks.com/help/gads/what-is-the-genetic-algorithm.html?searchHighlight=What%20is%20the%20Genetic%20Algorithm%3F.

[44] The MathWorks Inc., 2007, "Custom Data Type Optimization Using the Genetic Algorithm", from http://www.mathworks.com/help/gads/examples/custom-data-type-optimization-using-the-genetic-algorithm.html.

[45] The MathWorks Inc., 2006, "rand", from http://www.mathworks.com/help/matlab/ref/rand.html?searchHighlight=rand.

[46] University of California, Irvine – OpenCourseWare., 2011, "Generating random numbers: The rand() function", from http://ocw.uci.edu/upload/files/mae10_w2011_lecture13.pdf.

[47] Industrial Engineering & Management department, 2012-2015, "Center for Information Centric Engineering", Oklahoma State University, from http://vrice.okstate.edu/.

[48] Unity Technologies, 2015, "3D Formats", from http://docs.unity3d.com/Manual/3D-formats.html.

[49] 3D-Tool GmbH & Co. KG, 2015, "3D-Tool", from http://www.3d-tool.com.

[50] Siemens, 2015, "Solid Edge", from https://www.plm.automation.siemens.com/en_us/products/solid-edge/index.shtml.

[51] Unity Technologies, 2015, "Creating Scenes", from http://docs.unity3d.com/Manual/CreatingScenes.html.

[52] Unity Technologies, 2015, "GameObjects", from http://docs.unity3d.com/Manual/GameObjects.html.

[53] Geig, M., 2013. "Unity® Game Development." Sams Publishing, USA, Chap. 2.

[54] Unity Technologies, 2015, "Transforms", from http://docs.unity3d.com/Manual/Transforms.html.

[55] Geig, M., 2013. "Unity® Game Development." Sams Publishing, USA, pp. 41-42.

[56] Unity Technologies, 2015, "Asset Store", from http://docs.unity3d.com/Manual/AssetStore.html.

[57] Unity Technologies, 2015, "Positioning Game Objects", from http://docs.unity3d.com/Manual/PositioningGameObjects.html.

[58] Geig, M., 2013. "Unity® Game Development." Sams Publishing, USA, pp. 11, 33-34.

[59] Unity Technologies, 2015, "Path", from http://docs.unity3d.com/ScriptReference/Path.html.

[60] Unity Technologies, 2015, "Transform", from http://docs.unity3d.com/ScriptReference/Transform.html.

[61] Unity Technologies, 2015, "Array", from http://docs.unity3d.com/ScriptReference/Array.html.

[62] Unity Technologies, 2015, "Vector3", from http://docs.unity3d.com/ScriptReference/Vector3.html.

[63] Unity Technologies, 2015, "WaitForSeconds", from http://docs.unity3d.com/ScriptReference/WaitForSeconds.html.

[64] Unity Technologies, 2015, "MonoBehaviour.FixedUpdate()", from http://docs.unity3d.com/ScriptReference/MonoBehaviour.FixedUpdate.html.

[65] Unity Technologies, 2015, "Mathf.MoveTowards", from http://docs.unity3d.com/ScriptReference/Mathf.MoveTowards.html.

[66] Unity Technologies, 2015, "Time.deltaTime", from http://docs.unity3d.com/ScriptReference/Time-deltaTime.html.

[67] Press Release, 2008, "Virtual manufacturing technology a big factor behind ford's continually improving quality", Ford Motor Company, from http://ophelia.sdsu.edu:8080/ford/01-24-2009/about-ford/news-announcements/press-releases/press-releases-detail/pr-virtual-manufacturing-technology-a-27948.html.

[68] Unknown, 2012, "Ford creates virtual factory to simulate assembly-line process", The Engineer, from http://www.theengineer.co.uk/channels/design-engineering/news/ford-creates-virtual-factory-to-simulate-assembly-line-process/1013377.article.

[69] Luft, A., 2013, "Ford Begins Testing New Plant Virtualization Software From Siemens (With Video)", Motrolix, from http://motrolix.com/2013/12/ford-begins-testing-new-virtual-plant-software-from-siemens-with-video/.

[70] Burg, N., 2014, "Ford Pilots A New Virtual Assembly Plant To Build Its Cars", Forbes, from http://www.forbes.com/sites/unify/2014/03/31/ford-pilots-a-new-virtual-assembly-plant-to-build-its-cars/.

[71] Thornton, J., 2009, "At Ford, Ergonomics Meets Immersive Engineering", EHS Today, from http://ehstoday.com/health/ergonomics/ford-ergonomics-simulation-0409.

[72] Institute of Industrial Engineers, 2015 "Applied Ergonomics Conference", from http://www.iienet2.org/Ergo/Conference/.

[73] FordLightingLab, 2011, "Ford's Virtual Assembly Lines", from https://www.youtube.com/watch?v=ORuPe7r8IAI.

[74] FORD Car Center, 2011, "Ford Plans Virtual Factory for Ultimate Assembly Line Efficiency", from https://www.youtube.com/watch?v=3E4hv9qQU7w.

APPENDICES

Appendix A: MathWorks Permission

Authorization to use the MatLab codes from MathWorks in this thesis was requested on 4/24/2015 as follows:

Apr 24 (4 days ago)

Robledo Gallegos, Abraham <arobled@ostatemail.okstate.edu>

to MathWorks, arobled

Hello,

Thank you for receiving my call. As I explained I am a Master's Degree student presenting my work and Graduating this semester (May 2015) and before I upload my final report to the system on the deadline next week (April 30th-May 1st) I want to know how to correctly cite or refer to the following information you have available in your product help:

http://www.mathworks.com/help/gads/examples/custom-data-type-optimization-using-the-genetic-algorithm.html

I would also like to see if I may have your permission to place the codes that you make available in the link in my final report. I would like to explain that they were the starting point for me to be able to work on my specific project needs where I generated another custom type data for optimization as well. The scripts I refer to are the "Create_permutations", "Mutate_permutation" , "Crossover_permutation" and "Traveling_Salesman_Fitness".

Your script generates an open and random position sequence for the "cities" and I could not use them directly for my problem; however, they pointed me in the right direction to be able to use the Global Optimization Tool with the genetic algorithm solver to generate sequences (permutations) that I needed.

My thesis project is not related to MatLab; however, I use MatLab as a significant tool during my project. My thesis related to the construction of a virtual environment using a gaming engine. I use the sequence I get from MatLab to help validate the accuracy of my virtual environment.

I appreciate your guidance to appropriately report the material reference in the link above and I also will appreciate if you grant permission for me to use your codes as I explain the role of MatLab in my project.

Please submit your recommendations at your earliest convenience.

Best regards,

Abraham Robledo Gallegos

Oklahoma State University
Master's Degree Student, Industrial Engineering

Additional note,

I am already referencing the website regardless on my document:

http://www.mathworks.com/help/gads/examples/custom-data-type-optimization-using-the-genetic-algorithm.html

However, I wanted to reach out to you to see if I may have your permission to place the codes that you make available in the link in my final report.

Thanks again,

Abraham Robledo Gallegos

MathWorks response with Authorization to use the MatLab codes in this thesis was received on 4/27/2015 as follows:

MathWorks Customer Service via p2ihi0866lc2.i-ha1ueac.na15.bnc.salesforce.com 4:19 PM (9 hours ago)
to me

Hello Abraham,

The General Counsel of MathWorks put his matter on the top of his list today and said to say **"Used with the permission of MathWorks." He wanted me also to let you know that he really appreciates that you asked for permission.**

If you need further assistance regarding this request, please reply to this email preserving the Reference ID listed below. If you have a new Customer Support question, please submit a request here: http://www.mathworks.com/support/servicerequests/create.html

Thank you,

Simonne Bonfatti
MathWorks Customer Service
01336553
ref:_00Di0Ha1u._500i0OUwtn:ref

Appendix B: MatLab's Create Permutations script [44]

```
1.  function pop = create_permutations(NVARS,FitnessFcn,options)
2.  %CREATE_PERMUTATIONS Creates a population of permutations.
3.  %   POP = CREATE_PERMUTATION(NVARS,FITNESSFCN,OPTIONS)
    creates a population
4.  %  of permutations POP each with a length of NVARS.
5.  %
6.  %   The arguments to the function are
7.  %     NVARS: Number of variables
8.  %     FITNESSFCN: Fitness function
9.  %     OPTIONS: Options structure used by the GA

10. %   Copyright 2004-2007 The MathWorks, Inc.
11. %   $Revision: 1.1.6.1 $  $Date: 2009/08/29 08:27:32 $

12. totalPopulationSize = sum(options.PopulationSize);
13. n = NVARS;
14. pop = cell(totalPopulationSize,1);
15. for i = 1:totalPopulationSize
16. pop{i} = randperm(n);
17. end
```

Appendix C: Create Permutations [44] custom script

```
function pop = create_permutations_mod_2(NVARS,FitnessFcn,options)
%function calling similar to [44]
totalPopulationSize = sum(options.PopulationSize); %Population size definition similar to [44]
n = NVARS; %number of variables definition similar to [44]
pop = cell(totalPopulationSize,1);%CUSTOM -array with size equal to population size
pop2 = cell(totalPopulationSize,1);%CUSTOM - array with size equal to population size
for i = 1:totalPopulationSize  %cycle from 1 to population size
    pop2{i} = randperm(n-1)+1; %CUSTOM - fills permutations with all stations but station # 1; in an example with 7 cells-it goes from 2 to 7
    for j= 1:(n-1)   %CUSTOM - cycle from 1 to number of stations(n)-1
        pop{i}(1,j+1)=pop2{i}(1,j); %CUSTOM - starting in position 2 (j=1+1) pop fills with all stations (stored in pop2) except #1
    end
    pop{i,1}(1,1)=1 ;    % CUSTOM - all positions #1 are equal to station #1
    pop{i,1}(1,n+1)=pop{i,1}(1,1);     %CUSTOM -  added last position equal to the first position (1)
end
```

Appendix D: MatLab's Mutate Permutations script [44]

```
1.  function mutationChildren = mutate_permutation(parents ,options,NVARS, ...
2.  FitnessFcn, state, thisScore,thisPopulation,mutationRate)
3.  %   MUTATE_PERMUTATION Custom mutation function for traveling
    salesman.
4.  %   MUTATIONCHILDREN =
    MUTATE_PERMUTATION(PARENTS,OPTIONS,NVARS, ...
5.  %   FITNESSFCN,STATE,THISSCORE,THISPOPULATION,MUTATIONRATE)
    mutate the
6.  %   PARENTS to produce mutated children MUTATIONCHILDREN.
7.  %
8.  %   The arguments to the function are
9.  %     PARENTS: Parents chosen by the selection function
10. %     OPTIONS: Options structure created from GAOPTIMSET
11. %     NVARS: Number of variables
12. %     FITNESSFCN: Fitness function
13. %     STATE: State structure used by the GA solver
14. %     THISSCORE: Vector of scores of the current population
15. %     THISPOPULATION: Matrix of individuals in the current population
16. %     MUTATIONRATE: Rate of mutation

17. %   Copyright 2004 The MathWorks, Inc.
18. %   $Revision: 1.1.6.1 $  $Date: 2009/08/29 08:28:05 $

19. % Here we swap two elements of the permutation
20. mutationChildren = cell(length(parents),1);% Normally
    zeros(length(parents),NVARS);
21. for i=1:length(parents)
22. parent = thisPopulation{parents(i)}; % Normally thisPopulation(parents(i),:)
23. p = ceil(length(parent) * rand(1,2));
24. child = parent;
25. child(p(1)) = parent(p(2));
26. child(p(2)) = parent(p(1));
27. mutationChildren{i} = child; % Normally mutationChildren(i,:)
28. end
```

Appendix E: Mutate Permutations [44] custom script

- *function mutationChildren = mutate_permutation_mod(parents ,options,NVARS, FitnessFcn, state, thisScore,thisPopulation,mutationRate) %function calling similar to [44]*
- *n = NVARS; %number of variables definition similar to [44]*
- *mutationChildren = cell(length(parents),1 %Children group definition similar to [44]*
- *for i=1:length(parents) % For cycle similar to [44]*
    - *parent = thisPopulation{parents(i)}; % For cycle similar to [44]*
    - *parent(:,n+1)=[];% CUSTOM -removes the added last position (by create_permutations_mod) that moves the target back to the initial position at the end of the sequence*
    - *p = ceil((length(parent)) * ((1-(1/length(parent)))*rand (1,2)+(1/length(parent))));%CUSTOM - ensures that first station stays at beginning and end. Selects a position (station, other than the first one (and last one) to cross over/mutate. Based on the formula to generate values from the uniform distribution on the interval [a, b]:: r = a + (b-a).*rand().*
    - *child = parent; % Performs mutation [44]*
    - *child(p(1)) = parent(p(2)); % Performs mutation [44]*
    - *child(p(2)) = parent(p(1)); % Performs mutation [44]*
    - *mutationChildren{i} = child; % Performs mutation [44]*
    - *mutationChildren{i,1}(1,n+1)= mutationChildren{i,1}(1,1);%CUSTOM-adds to each child sequence one last position with the initial station (1)where the target returns at the end of the sequence*
- *end*

Appendix F: MatLab's Crossover Permutations script [44]

```
1.  function xoverKids  = crossover_permutation(parents,options,NVARS, ...
2.  FitnessFcn,thisScore,thisPopulation)
3.  %   CROSSOVER_PERMUTATION Custom crossover function for traveling
    salesman.
4.  %   XOVERKIDS =
    CROSSOVER_PERMUTATION(PARENTS,OPTIONS,NVARS, ...
5.  %   FITNESSFCN,THISSCORE,THISPOPULATION) crossovers PARENTS to
    produce
6.  %   the children XOVERKIDS.
7.  %
8.  %   The arguments to the function are
9.  %     PARENTS: Parents chosen by the selection function
10. %     OPTIONS: Options structure created from GAOPTIMSET
11. %     NVARS: Number of variables
12. %     FITNESSFCN: Fitness function
13. %     STATE: State structure used by the GA solver
14. %     THISSCORE: Vector of scores of the current population
15. %     THISPOPULATION: Matrix of individuals in the current population

16. %   Copyright 2004 The MathWorks, Inc.
17. %   $Revision: 1.1.6.1 $  $Date: 2009/08/29 08:27:33 $

18. nKids = length(parents)/2;
19. xoverKids = cell(nKids,1); % Normally zeros(nKids,NVARS);
20. index = 1;

21. for i=1:nKids
22. % here is where the special knowledge that the population is a cell
23. % array is used. Normally, this would be thisPopulation(parents(index),:);
24. parent = thisPopulation{parents(index)};
25. index = index + 2;

26. % Flip a section of parent1.
27. p1 = ceil((length(parent) -1) * rand);
28. p2 = p1 + ceil((length(parent) - p1- 1) * rand);
29. child = parent;
30. child(p1:p2) = fliplr(child(p1:p2));
31. xoverKids{i} = child; % Normally, xoverKids(i,:);
32. end
```

Appendix G: Crossover Permutations [44] custom script

- *function xoverKids = crossover_permutation_mod(parents,options,NVARS, FitnessFcn,thisScore,thisPopulation) %function calling similar to [44]*
- *n = NVARS; %number of variables definition similar to [44]*
- *nKids = length(parents)/2; %Children length definition similar to [44]*
- *xoverKids = cell(nKids,1); %Children group definition similar to [44]*
- *index = 1; %For cycle similar to [44]*
- *for i=1:nKids %For cycle similar to [44]*
- *parent = thisPopulation{parents(index)}; %For cycle similar to [44]*
- *index = index + 2; %For cycle similar to [44]*
- *parent(:,n+1)=[];%CUSTOM- removes the added last position (by create_permutations_mod) that moves the target back to the initial position at the end of the sequence*
- *p1 = ceil((length(parent) -1) * ((1-(1/(length(parent)-1)))*rand+ (1/(length(parent)-1)))); %CUSTOM - ensures that first station stays at beginning and end. Selects a position (station, other than the first one (and last one) to cross over/mutate. %Based on the formula to generate values from the uniform distribution on the interval [a,b]:: r = a + (b-a).*rand().*
- *p2 = p1 + ceil((length(parent) - p1- 1) * rand); % Performs crossover [44]*
- *child = parent; % Performs crossover [44]*
- *child(p1:p2) = fliplr(child(p1:p2)); % Performs crossover [44]*
- *xoverKids{i} = child; % Performs crossover [44]*
- *xoverKids{i,1}(1,n+1)= xoverKids{i,1}(1,1);% CUSTOM - adds to each child sequence one last position with the initial station (1) where the target returns at the end of the sequence*
- *end*

Appendix H: MatLab's Traveling Salesman Fitness script [44]

```
1.  function scores = traveling_salesman_fitness(x,distances)
2.  %TRAVELING_SALESMAN_FITNESS  Custom fitness function for TSP.
3.  %   SCORES = TRAVELING_SALESMAN_FITNESS(X,DISTANCES) Calculate
    the fitness
4.  %   of an individual. The fitness is the total distance traveled for an
5.  %   ordered set of cities in X. DISTANCE(A,B) is the distance from the city
6.  %   A to the city B.

7.  %   Copyright 2004-2007 The MathWorks, Inc.
8.  %   $Revision: 1.1.6.1 $  $Date: 2009/08/29 08:28:36 $

9.  scores = zeros(size(x,1),1);
10. for j = 1:size(x,1)
11. % here is where the special knowledge that the population is a cell
12. % array is used. Normally, this would be pop(j,:);
13. p = x{j};
14. f = distances(p(end),p(1));
15. for i = 2:length(p)
16. f = f + distances(p(i-1),p(i));
17. end
18. scores(j) = f;
19. end
```

Appendix I: Script created in Unity® to simulate near optimal sequences in the virtual environment

```
1.  import System;        //Initializing code
2.  import System.IO;     //Initializing code
3.  var target : Transform;  //object to be moved
4.  var Initial: Transform;  //object to be the initial place from where the target object starts
    moving
5.  var Sequence: Transform[]= new Transform[31]; // array that represents the sequence of
    stations where target moves
6.  var speed : float; //Variable that defines how fast the target moves
7.  var i = 0; //Counter for the "for" cycle that moves the target from one station to another
8.  var vect: Vector3; //vector definition
9.  private var dist : float; //Variable used to calculate the distance traveled on each
    sequence segment
10. private var Totdist : float;//Variable used to calculate the overall cumulative distance
11. target.position = Initial.position; //Moves target to the initial position
12. print("Initial coordinates: "+Initial.position);//Prints initial position coordinates in the
    Console window
35. for (i=0; i < Sequence.Length; i++)//cycle that moves target until sequence is completed
36. {
37. dist = Vector3.Distance(Sequence[i].position,target.position);//Calculates segment
    distance traveled by target
38. Totdist=Totdist+dist;//Calculates cumulative traveled distance
39. vect = Sequence[i].position; //Redefines vect as next (i) sequence position
40. yield WaitForSeconds((dist/speed)+1);//Introduces a pause for target to stop at each
    station in sequence
41. print("New coordinates: "+target.position); //Prints coordinates for each station where
    the target stops by
42. print("Segment distance: "+dist);      //Prints segment distance value
43. print("Cumulative distance: "+Totdist);//Prints cumulative distance value
44. print(vect);//Prints coordinates of sequence object where target arrives into
45. }
46. function FixedUpdate() {//Function that constantly updates the status of the model (and
    target)
47. target.position = //New position of target equals:
48. Vector3(Mathf.MoveTowards(target.position.x,vect.x,speed*Time.deltaTime),//moving in
    X at speed rate
49. Mathf.MoveTowards(target.position.y,vect.y,speed*Time.deltaTime), //moving in Y at a
    speed rate
50. Mathf.MoveTowards(target.position.z,vect.z,speed*Time.deltaTime));//moving in Z at a
    speed rate
51. }
```

VITA

Abraham Robledo Gallegos

Candidate for the Degree of

Master of Science

Thesis: DEVELOPMENT OF A VIRTUAL FACTORY ENVIRONMENT TO STUDY, SIMULATE AND IMPROVE THE MATERIAL FLOW BETWEEN MULTIPLE MICRO ASSEMBLY WORK CELLS

Major Field: Industrial Engineering & Management

Biographical:

Education:

Completed the requirements for the Master of Science in Industrial Engineering & Management at Oklahoma State University, Stillwater, Oklahoma in May, 2015
Completed the requirements for the Bachelor of Science in Industrial Engineering at Universidad de las Americas, Puebla, Puebla / Mexico in 2007.

Experience:

Lennox International- Heatcraft Worldwide Refrigeration *Columbus, GA*
Environmental, Health and Safety Manager *Dec 11 –Current*
- *Awards & Achievements:*
  - LII Safety Pacesetter Award winning facility *2014 & 2015*
  - LII Mentoring for Diversity Leadership Program Graduate *Mar 2014*
  - Heatcraft Emerging Leader Development Program Graduate *Mar 2015*

Lennox International- Heatcraft Worldwide Refrigeration *Tifton, GA*
Manufacturing Engineer *Jul 07 - Dec 11*
Cadbury ADAMS *Puebla, Mexico*
Thesis on simulation & improvement of production lines *Aug 06- May 07*
Lennox International- Allied Air Enterprises *Blackville, SC*
Engineering Ergonomics Intern *May 06 - Aug 06*

Professional Memberships:

IIE Applied Ergonomics Conference program committee member, speaker and attendant from 2007 to 2015.