Proceedings of the 1993 Winter Simulation Conference
G.W. Evans, M. Mollaghasemi, E.C. Russell, W.E. Biles (eds.)

QUEST - QUEUEING EVENT SIMULATION TOOL

Samir K. Mahajan
Scott K. Brewer
Christopher D. Bien

Deneb Robotics, Incorporated
Auburn Hills, Michigan 48321-4687, U.S.A.

ABSTRACT

QUEST (QUeueing Event Simulation Tool) is a 3D graphics simulation tool used to analyze and visualize complex manufacturing and material handling systems. QUEST provides a complete solution for all aspects of manufacturing planning from the evaluation of strategies and plant floor layout to the programming of automation equipment. This simulation technology can be applied to flexible manufacturing systems (FMS), Just-In-Time (JIT), business re-engineering, team labor, cost, and a host of other issues facing manufacturing. Covered in this presentation will be the unique features of QUEST and two tutorials to illustrate its scope.

1 QUEST - AN OVERVIEW

1.1 Simulation

QUEST is a hybrid discrete event simulation tool, which enables both object oriented, as well as simulation language approaches to modeling. With this powerful feature, QUEST not only takes advantage of the ease of use and rapid model developing capabilities of object oriented modeling, but also provides the depth and extendibility of language-based behavior modeling. QUEST sessions may be either animated (i.e., model building), or non-animated (i.e., when collecting statistical results). Other options available within the simulation mode include warm up times, multiple random seeds, animation toggling, statistics toggling, strip charts, and status highlighting.

1.2 Visualization

QUEST is a three-dimensional modeling tool, which interactively combines the simulation and animation paradigms found in discrete event simulation tools. This allows simulation characteristics, such as conveyor lengths, distances between resources, and widget characteristics to be automatically defined through model geometry and layout.

Realistic visual representations of virtually any system can be achieved with QUEST's true three-dimensional modeling and fast rendering capabilities. Compared to the quality of existing two-dimensional animation, the three-dimensional visualization of QUEST facilitates both perspective and orthographic projections. The use of high performance graphics allows interactive navigation through the model for rapid user comprehension during model construction, and appreciation by audiences that are not simulation experts.

In industry today, two-dimensional animation is more prevalent than three-dimensional modeling. This phenomenon can be attributed primarily to the limited number of commercially available, fully functional, three-dimensional discrete event packages and the feasibility of purchasing advanced hardware. As the cost for computer workstations continues to decrease, while their computational speed improves, three-dimensional tools are becoming more accessible for simulating manufacturing, material handling and other real world systems.

2. QUEST MENU SYSTEM

QUEST's patented* three-tier menu system provides a mouse-driven interface through which every command is no more than two mouse clicks away. There are seven main components of the QUEST menu system, they are referred to as contexts. Through use of these menus and QUEST's intelligent popup system, object-oriented model construction and interrogation is possible. The direct object manipulation approach offered by the

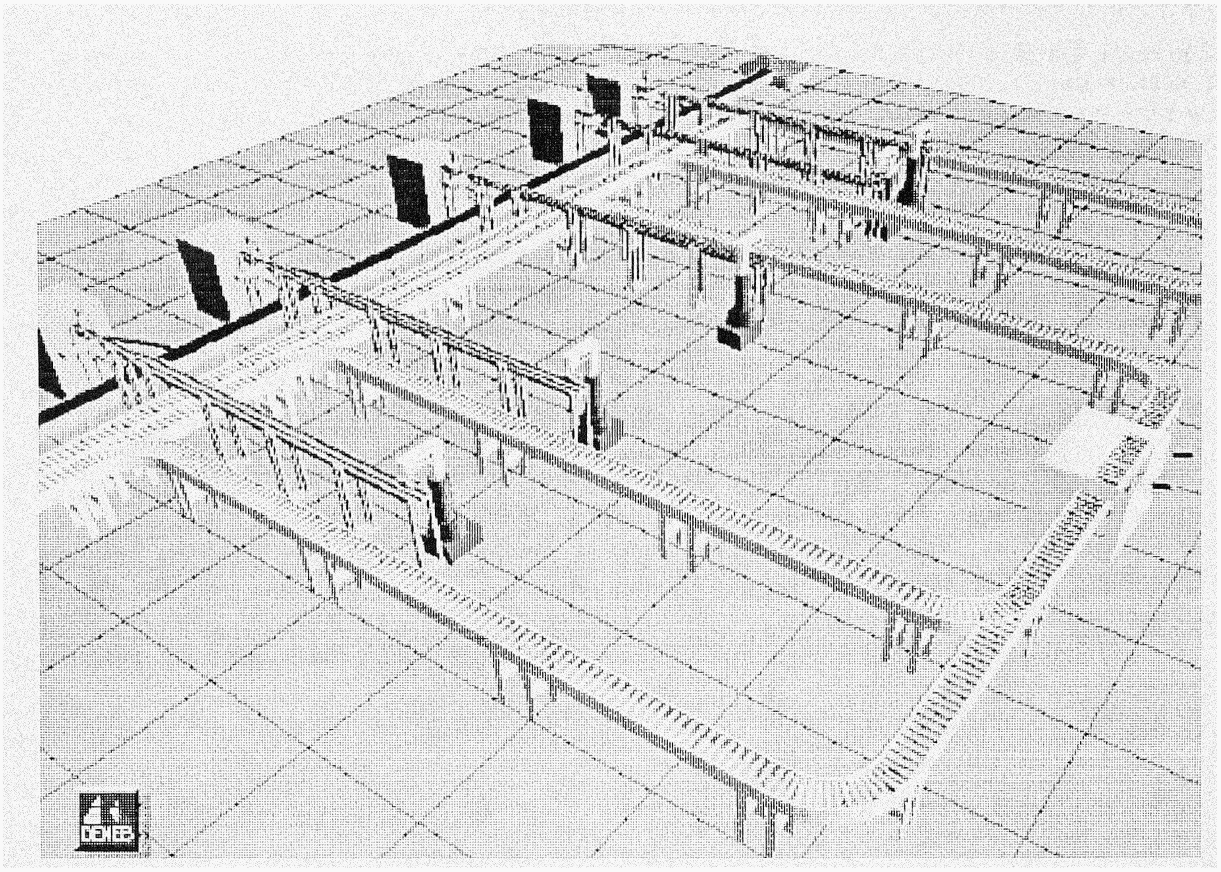


Figure 1 : A Sample QUEST Layout

menus in combination with interactive visualization results in rapid model building and prototyping.

2.1 CAD Context

Allows users to create and store three-dimensional visual representations of objects, such as widgets, workcells, transports, etc. The CAD system is an easy to use, self-sufficient package that provides an interface with other CAD systems through production proven, direct data translators;

-IGES
-DXF
-VDA
-CATIA
-WAVEFRONT

2.2 Model Context

QUEST provides a built-in library containing a rich set of production resources that are the fundamental building blocks of manufacturing processes. In addition, the library is user extendible to accomodate for specific modeling tasks. QUEST enables users to define resource characteristics and the interrelationships among them. It also allows the user to define widgets, representing the material or parts that flow through the system.

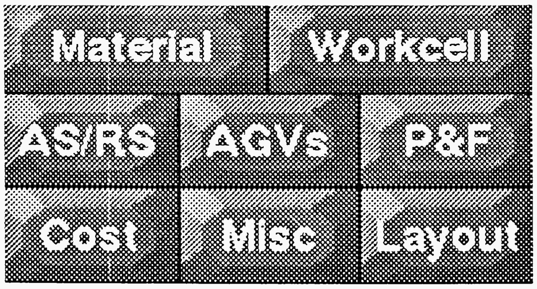


Figure 2 : Model Page Layout

2.3 Run Context

Permits users to execute simulations, procure statistic collection control and data set management, utilize event tracing for debugging, examine single run statistics, and create graphs and charts illustrating statistical results.

2.2.4 Statistics Context

Provides a facility for comparing different simulation runs and creating graphs to illustrate the results. The functions automatically label and title the graphs being created. Other functions are provided that allow the user to customize graph construction and manipulation on a quantity-to-quantity basis.

2.5 User Context

Allows user-defined menu buttons and customized interfaces. Included under the User page are User buttons and Macro buttons. User buttons allow personal arrangement of the most frequently used buttons. Macro buttons allow users to tie Simulation Control Language (SCL) or Batch Control Language (BCL) programs into the QUEST menu system for easy accessibility.

2.6 Analysis Context

Provides a host of functions for analyzing the physical state and the logical construction of a factory model. It assists the user in identifying characteristics of simulations, such as resource and widget characteristics, X-Y-Z distances between components, as well as logic considerations like buffer disciplines and routing logics.

2.7 System Context

The System context offers necessary functions in the areas of data files, user-configuration, utilities, and other system issues. It gives the user the ability to define system attributes including model view characteristics (i.e., lighting, floor grid, floor and background color, hard copy printouts and annotations). A menu-driven UNIX file interface is also available, including configuration files, project data management, and user configuration.

3 PROGRAMMING CAPABILITIES

3.1 Simulation Control Language (SCL)

The Simulation Control Language (SCL) is a high level procedural language for satisfying unique modeling requirements necessary for studying specific systems. SCL is embedded in a simulation model by associating modular procedural programs with specific resource entities to govern the behavior of that resource. These distributed logics are triggered by specific events occurring during the simulation. Examples of SCL logic triggers are routing, processing, queueing, and so on. Through SCL, users can develop custom heuristics and decision policies. Advanced applications can utilize SCL's UNIX file and socket interfaces to connect in real-time to other programs in a network.

3.2 Batch Control Language (BCL)

The Batch Control Language (BCL) is a powerful communication, command, and control system for accessing and operating QUEST. Furthermore, the BCL system may be accessed from either *inside* or *outside* the QUEST menu system:

Inside the QUEST menu system, BCL offers alternatives to the point and click approach through Macro buttons and text commands. 'SET' commands, for instance, can be used to set resource and widget characteristics, logic options, world views and simulation run parameters. Another form of context commands are INQUIRE commands. These can be executed to receive statistical results on resources and widgets.

Outside the QUEST menu system, BCL performs in two different modes; stream and socket. In both cases, the QUEST window appears without menus. The stream mode allows a text file to be used as input and can direct output to an output file. For example:

```
quest -b <batch_file >output_file
```

Under the stream mode, any UNIX stream may be directed to QUEST. The socket mode allows any external program, residing on any machine in a TCP/IP network, to invoke and communicate with QUEST through a socket, using BCL commands and return codes. In this way, user applications may systematically conduct experiments and optimize simulation parameters.

4 QUEST - Special Features

Under the QUEST model building section, standard resources include buffers, conveyors, and workcells. Workcells conduct processes that operate on widgets, possibly transforming them into a new type of widget. Buffers store widgets according to a variety of removal and queueing logics. Conveyors transport widgets from one location to the other. Some of the unique features that QUEST offers to the user include:

4.1 Power and Free Systems

This is suitable for any system consisting of tracks and carriers, such as mono-rail systems and synchronous indexing types of machinery. The system contains physically based features such as dog-delays, merge, and clearances.

4.2 Automatic Guided Vehicle (AGV)

The AGV system in QUEST is comprised of various AGVs, a network of control points and paths, and global AGV controllers. The controllers are responsible for dispatching idle AGVs in response to requests to transport widgets.

4.3 Automatic Storage and Retrieval Systems (AS/RS)

The AS/RS is comprised of aisles, racks, and bins. Arrival and departure stations govern the storage and retrieval of widgets from these locations.

4.4 Labor

Labor in QUEST is explicitly modeled with a Labor Resource which service workcell's processes. Labor can be shared among several workcells, and labor teams can be made to work together. Statistics such as move time, idle time, and utilization are collected.

4.5 Reports

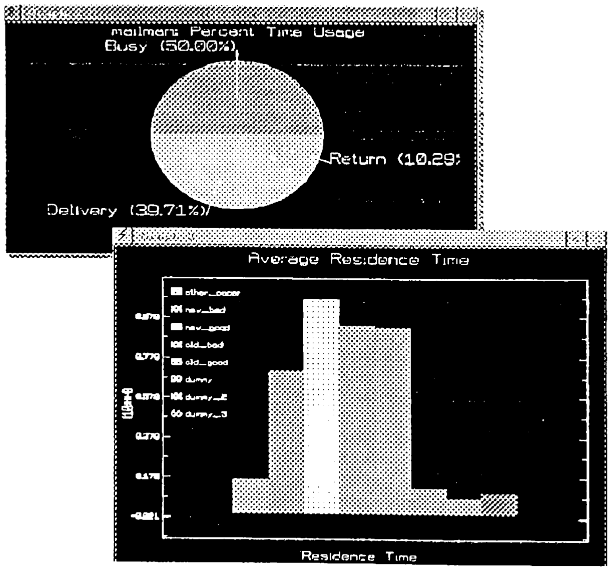


Figure 3 : Multiple Graphs

Graphs, bar charts, pie charts, dynamically interactive strip charts, and custom read/write ASCII file based reports can be created through QUEST. These reports plot useful statistics covering resource utilization, widget throughput time, average buffer length, etc. Reports can also be written to data files which can be printed in report form without additional manipulation from the user.

4.6 Activity Based Costing (ABC)

QUEST provides all cost attributes necessary to conduct explicit cost analysis for calculating and studying associated production costs. Cost attributes may be customized to reflect the cost attributes for specific models (i.e., overhead, breakdown and repair, labor, materials, etc.).

4.7 Push and Pull Production Methodology

In QUEST, an explicit model for request information flow can be represented through Push and Pull production methodologies. These methods can be integrated to support the delicate balance of push and pull methods found in many manufacturing systems.

4.8 External Communication

Through the Simulation Control Language (SCL) and the Batch Control Language (BCL), QUEST can send and receive values to communicate with any external process or machine. For example, BCL can be invoked to open UNIX sockets to run multiple copies of QUEST on different workstations. With this functionality, users can simultaneously conduct design of experiments (DOE), such as hill climbing algorithms to optimize parameters between model runs.

4.9 Integrated Simulation and Animation

With QUEST, total interactive visualization of simulation and geometric models allow execution through *incremental compilation*. Also, the user has the flexibility to debug, test, analyze statistics, experiment with multiple runs, and interactively validate models without recompiling or waiting for the entire program scenario to finish.

5 TUTORIALS

This next section will illustrate the steps needed to create two demonstrations through QUEST.

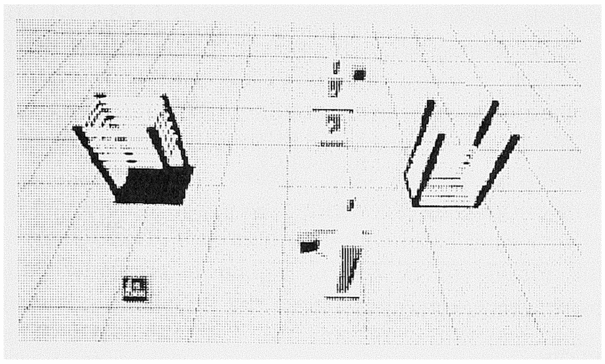
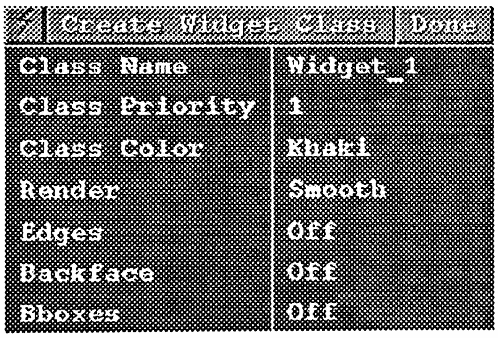


Figure 4: Tutorial Layout
From left to right : source, buffer, two workcells & sink

5.1 Demonstration 1 - Basic Model

In this demonstration, a simple simulation for two different types of raw materials passing through a work station is created. The raw material will travel to either of two workcells depending on its class and the workcell's requirements.

The first step is to create two different classes of widgets which represent raw materials (refer to figure 5.1).



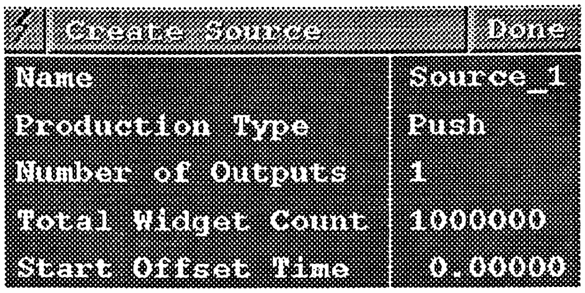
Class Name	Widget_1
Class Priority	1
Class Color	Khaki
Render	Smooth
Edges	Off
Backface	Off
Bboxes	Off

Figure 4.1 : Widget Creation

Both widgets will be made according to the table shown; Widget_2 will be given a different class color (i.e., chocolate) so that it can be easily distinguished.

Next, a source to produce the widgets will be created (refer to figure 5.2).

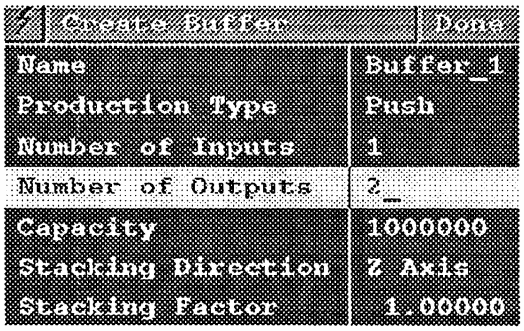
The push source will begin creating widgets at time zero and will make up to 1,000,000 widgets. The Inter-Arrival-Time (IAT) for the source can be changed to any built-in distribution or may be user-defined. This model's (IAT) is an exponential distribution with a mean of ten seconds. The source's widget fractions should also be changed so that the source creates widgets at a ratio of 1:1. QUEST will automatically normalize this ratio into a probability of 0.5 for each.



Name	Source_1
Production Type	Push
Number of Outputs	1
Total Widget Count	1000000
Start Offset Time	0.00000

Figure 4.2 : Source Creation

A buffer with one input and two outputs is created next (refer to figure 5.3).



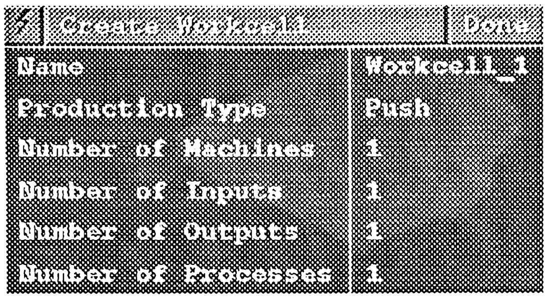
Name	Buffer_1
Production Type	Push
Number of Inputs	1
Number of Outputs	2_
Capacity	1000000
Stacking Direction	Z Axis
Stacking Factor	1.00000

Figure 4.3 : Buffer Creation

The input for the buffer will be the source and the output will be the two workcells. It will have an infinite capacity and will stack widgets in the Z-axis. The buffer is used so that if either of the workcells are busy, the widgets will 'wait' in the buffer. This prevents the source from being 'blocked' and inhibited from creating widgets. The buffer's removal logic will be changed to 'required widgets'. This allows the workcell to take the widget class it requires.

Next we will create two workcells (refer to figure 5.4).

Both of the push production workcells will have one process; the second of the workcells will have two sub-processes. Cycle times for the workcells (on a sub-process basis) may be changed to any built-in or user-defined distribution. This model uses a normal distribution for Workcell_1 with a mean of fifteen seconds and standard deviation of two seconds. Workcell_2 uses a constant value of ten seconds for sub-process one and a uniform distribution between five and ten seconds for sub-process two. Next, we will set the required widgets for each workcell.

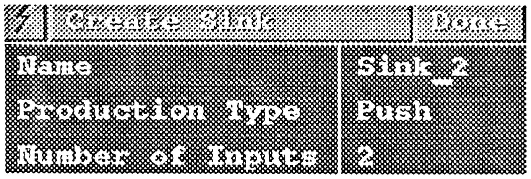


Create Workcell	Done
Name	Workcell_1
Production Type	Push
Number of Machines	1
Number of Inputs	1
Number of Outputs	1
Number of Processes	1

Figure 4.4 : Workcell Creation

Since the workcells have only one process, the process logic will not need to be changed. If there had been multiple processes, the process logic could have been set through built-in procedures, or through SCL.

A push production type sink that will accept the finished product of both workcells will now be created (refer to Figure 5.5). This sink will have two inputs, one from each of the two workcells. Finally, all the resources need to be connected together in a logical fashion (i.e., the source's output and the buffer's input should be connected, etc.).



Create Sink	Done
Name	Sink_2
Production Type	Push
Number of Inputs	2

Figure 4.5 : Sink Creation

The simulation is run for 1000 seconds. Run statistics, such as resource utilization and widget throughput times, can be viewed during or after the completion of the simulation.

5.2 Demonstration 2 - With SCL

The objective of this tutorial is to demonstrate the user's ability to control a model through SCL logic. This model uses the first model as the basis, but adds additional complexity; both workcells will begin processing at the same time.

SCL code will be written as follows:

```
-- GLOBAL DECLARATIONS

VAR

BUFFER_1            : RESOURCE
WORK_1              : RESOURCE
WORK_2              : RESOURCE
WID_KHAKI_1         : WIDGET
WID_CHOCOLATE_1     : WIDGET

------------------------------------------------------

PROCEDURE INITIALIZE()

BEGIN

     BUFFER_1  = GET_RESOURCE ('Buffer_1')
     WORK_1    = GET_RESOURCE ('Workcell_1')
     WORK_2    = GET_RESOURCE ('Workcell_2')

END

------------------------------------------------------
PROCEDURE BUFFER_ROUTE_LOGIC()

VAR
 ii          : INTEGER -- incrementer
 cur_wid     : WIDGET -- current widget

BEGIN

 IF ((WORK_1->status == idle) AND (WORK_2->status == idle))
 THEN

     WID_KHAKI_1 = NULL
     WID_CHOCOLATE_1 = NULL

     FOR ii = 1 TO cres->num_widgets DO
        cur_wid = cres -> widgets[ii]

        IF (( WID_KHAKI_1 == NULL ) AND
             ( cur_wid-> class_name == 'Khaki')) THEN
                 WID_KHAKI_1 = cur_wid
        ENDIF

        IF (( WID_CHOCOLATE_1 == NULL ) AND
             ( cur_wid-> class_name == 'Chocolate')) THEN
                 WID_CHOCOLATE_1 = cur_wid
        ENDIF

     ENDFOR

   IF (( WID_KHAKI_1 <> NULL ) AND
        ( WID_CHOCOLATE_1 <> NULL )) THEN
             ROUTE(WID_KHAKI_1,1)
             ROUTE(WID_CHOCOLATE_1,2)
   ENDIF
 ENDIF
END
```

The initialized procedure will run before the model starts its simulation. This initializes the pointers WORK_1, WORK_2, and BUFFER_1 to resources in the models. In the buffer route logic, check to see if both workcells are idle. If they are, check to make sure the buffer contains at least one widget of each class. This is done by running through the buffer's contents. If

WID_KHAKI_1 is currently pointing to NULL, and the current widget is of that class, then the pointer is set to the current widget. Likewise, the same is done for WID_CHOCOLATE_1. If there is at least one widget of each type in the buffer and both workcells are idle, then route the first khaki widget (WID_KHAKI_1) and the first chocolate widget (WID_CHOCOLATE_1).

In QUEST, this model's buffer route logic will be called whenever a widget arrives in the buffer and whenever either of the workcells become idle (and backfires to the buffer). After the SCL code is assigned to the buffer's route logic, the code replaces the standard route logic of the buffer.

The simulation is run for 1000 seconds. Run statistics, such as resource utilization and widget throughput times, can be viewed during or after the completion of the simulation.

6 CONCLUSION

QUEST is endowed with a powerful architecture and feature set which allows it to address modern manufacturing problems such as adaptive systems, JIT planning, business re-engineering, flexible conveyors, labor issues, and cost analysis. It also firmly establishes accurate, three-dimensional physical modeling and graphics as an essential ingredient in manufacturing simulation.

QUEST is equipped with production-proven features and flexibility required for serious simulation environments involving teamwork, data connectivity, and distributed computing techniques. For all of its dynamic functionality and power, QUEST also delivers ease of use, simplicity, and efficiency required by its users.

ACKNOWLEDGMENTS

The authors wish to thank Charles E. Fuller for his contribution to this paper.

AUTHOR BIOGRAPHY

SAMIR K. MAHAJAN completed a summer internship program at Deneb Robotics, Inc. He is currently persuing a B.S. in Systems Engineering at the University of Virginia. Since joining Deneb, he has learned the QUEST and IGRIP systems and has modeled manufacturing and office simulations.

SCOTT K. BREWER is an Applications Engineer with Deneb Robotics. He received his B.S. degree from Hope College, his MSOR from Wayne State University, and his MBA from the University of Houston. He is also a member of the Institute of Industrial Engineers.

CHRISTOPHER D. BIEN is a marketing and sales representative with Deneb Robotics. He received his B.A. degree in marketing from Michigan State University.

* U.S. Patent No. 5,179,653