2007

Virtual design for the interactive placement of baffles in air flow

Jared Mark Abodeely
Iowa State University

Virtual design for the interactive placement of baffles in air flow

by

Jared Mark Abodeely

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Mechanical Engineering

Program of Study Committee:
K. M. Bryden (Major Professor)
Eliot Winer
Tom Shih

Iowa State University

Ames, Iowa

2007

TABLE OF CONTENTS

ABSTRACT

This thesis develops an interactive engineering design tool for the placement of baffles in air systems. This design tool integrates together the needed design analysis tools into a single interactive environment. This design tool has three primary components: (1) a CFD model of air flow through the system, (2) a virtual environment, and (3) a baffle placement and remeshing scheme. The CFD model used in conjunction with the design tool should be fast, accurate, and answer the current questions associated with the air system. The virtual environment includes an intuitive user interface and a graphical representation of the model. The design tool should also incorporate an easy baffle placement and remeshing scheme to easily allow the creation new CFD models for analysis. This allows users to design baffle configurations in a virtual environment and complete analysis through the integration of analysis tools through this single design tool. In the work presented in this thesis the design tool is demonstrated on two examples: an engine platform and a cleaning shoe in a combine. Current methods for design and decision-making processes for baffle placement within these types of air systems are inefficient and time consuming. Often CFD models of air systems are large and complex and require several days to complete an analysis. Under the current process, analysts are required to create new full-scale CFD models for each design iteration. Designers then have to wait for the results to come back before determining if another modification to the air system design is necessary. This can be extremely time consuming if a suitable solution is not immediately found. Introduction of the design tool developed in this thesis into the design and decision-making process simplifies and improves the overall process and the roles of those involved.

CHAPTER 1: INTRODUCTION

A self-propelled grain harvester is a complicated piece of machinery that is constructed from several complex systems, including the cutting system, the threshing system, and the separation system. Together, these systems move harvested material from the field to the clean grain bin. Because of the separation system's complexity, design changes are typically derived from the engineer's knowledge and experience with the system. With so many variables and components affecting the machine's efficiency, completely understanding the air flow system and the impact of one design change on the entire system is difficult. This thesis examines how computational fluid dynamics (CFD), when introduced into the design process, allows engineers to gain a better understanding of the air flow within the separation system and the impact of design changes on the system.

The ability to manage air movement within harvesting equipment is beneficial in many respects. Air is used to separate the grain and chaff and to transport the chaff out of the system. Generally, fans are the main source of air movement in harvesting equipment, but limited space and power requirements hinder the number of fans in the system, the location of the fans, and the power provided to the fans. This means that some locations within the harvesting equipment may not get sufficient air movement, which could lead to decreased machine performance. Improving air flow within the system requires baffles to redirect the air to specific areas where performance metrics are not met. Often, finding the ideal location or orientation of a baffle is time-consuming and expensive due to the nature of field testing and current analysis methods.

Typically, the design and analysis of a system is cyclical due to the nature and current capabilities of the design and analytical software. Current design software such as computer-

aided design (CAD) is used to create and assemble system components. Similarly, analysis software provides the means to determine fluid flow, stress, and other key system properties. However, the interface between the CAD software and various analysis packages is fully integrated and does not readily lend itself to complex component and system design. Currently, obtaining analytical results requires the designer to provide CAD models of the system to the CFD analyst. The analyst uses the model to create a mesh within CFD software to simulate fluid moving around and through the system. Upon completion of the simulation, analysts inform designers of the results through plots or basic visualization tools. Discussion and suggestions may ensue, leading the designer to return to the CAD model for subsequent design changes. This cycle continues until a design meets the requirements established at the beginning of the project. This process is time-consuming and inefficient. This paper develops an alternative process for the design of harvester air systems that addresses design issues earlier and in a parallel, real-time, interactive framework in an effort to save resources over the duration of the design process.

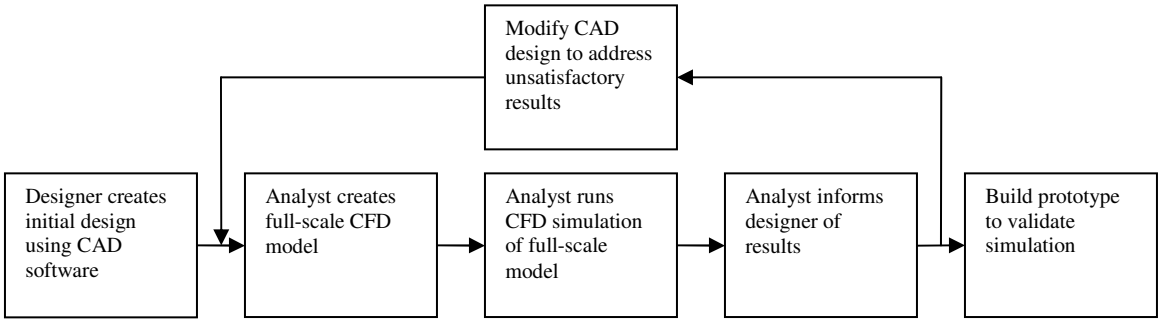

Figure 1. Current design process

Early in the design of a system, engineers look for direction in their decision-making process. Often this direction comes from product analysis, which can be time-consuming due to the level of accuracy required by the designers and analysts. If quicker analysis

models were created at a lower yet acceptable level of accuracy, many design avenues could be explored in a relatively short amount of time. This would enable the decision makers to get a better sense of direction for the product earlier in the design process. Establishing a clear direction for a product design early in the process reduces the time associated with investigating insufficient full-scale models.

Virtual engineering is a relatively new concept that is being applied to the design and decision-making process in the proposed work. Virtual engineering is a new technology that integrates geometric models with other engineering tools such as CFD, FEA, Excel, etc., in a computer-generated environment. One goal of virtual engineering is to create a decision-making environment that allows design changes within a virtual environment and that returns near-real-time solutions (Huang et al., 2004). The application of virtual engineering to the design and decision-making process can decrease the time, expense, and personnel required in the process.

The proposed work in this thesis is the development of an interactive baffle placement tool that integrates design and analysis tools into a single, collective environment. The tool will allow designers to create baffle configurations and complete analysis in a timely manner. Reduced CFD models will be utilized for their low simulation runtimes. The tool is built upon a virtual engineering software framework known as VE-Suite, an open-source effort by the Virtual Engineering Research Group at Iowa State University. The goal of this research is to address current design issues associated with air flow systems in harvesting equipment through the development of an interactive baffle design tool using virtual engineering concepts that will reduce time, money, and resources needed to find effective baffle configurations.

CHAPTER 2: BACKGROUND

Engineering is the process of making a series of decisions about complicated, complex, and uncertain systems, components, and processes. The methods and procedures that engineers use to make decisions have evolved over the last half century. Engineering methods must be enhanced to improve product development, increasing production and decreasing cost through utilization of the last few decades' technological advances as well as the changes in management and decision-making methodology. The introduction of computers into the design and analysis processes has vastly improved the product development cycle from conception to production. As technology continues to advance, models and simulations of larger, more complex problems can be addressed. The ability to effectively create and analyze a product in a computer-generated environment can reduce both cost and time in making engineering decisions (Cao et al., 2004). There is an ongoing effort to create computer-generated environments that addresses designers' needs of managing complex models while maintaining the goals of low cost and production time.

2.1 Engineering Decision Making

A design engineer is driven by expectations to create a product that meets the consumer's needs and is also safe and cost-effective. While these goals have not changed over the course of the last century, the methods by which they are accomplished has changed. In the past, the engineering decision-making process was a relatively linear course of action. An engineer with a specific skill set would complete his or her task on the product before it could be passed down the line to the next engineers to allow them to perform their tasks. This methodology allows for minimal interaction or deviance within the process, which can lead to complications when changes or modifications are introduced at a later design stage.

Many organizations have been restructuring into cross-functional, multi-disciplinary units to improve their design process (Krishnan et al., 1997), which has lead to a more efficient and parallelized decision-making process with many benefits.

As the complexity of systems continuously increases along with the complexity of the socio-economic environment, considering all aspects of a problem becomes more difficult for a single designer (Kim et al., 1999). By establishing group decision making within the design process, more complexity can be and often is introduced into the process. Group decision making also allows interaction between designers with different backgrounds and ideas to discuss product direction and purpose. The key to group decision making is to reach a consensus about the product that will have the greatest benefit for the company. Experts from different fields coming together to collaboratively and simultaneously work on a product to improve the design, design process, or decision-making process is known as *concurrent engineering*. Concurrent engineering is being used by many companies and has resulted in companies providing better products to the market more quickly (Haque et al., 2000). This sort of interaction among designers can influence new ideas and directions both on the individual level as well as the group level.

(Coles et al., 2005) states that three factors are needed in the decision-making process: knowledge, skills, and values. Although random, uneducated attempts at the decision-making process may yield a positive solution, this should not be the path by which decisions are made. Design engineers should utilize their knowledge and skills to formulate a well-thought-out, quality product. Knowledge must not only be drawn from their analysis tools, but must also include their observations and experience from previous product design. Constructively utilizing their skills based on their knowledge can greatly benefit product

design. These concepts are important when creating a virtual environment. Knowing the questions that need to be answered as well as the available resources allows the engineer to create a virtual environment that is tailored to answer these questions.

2.2 Development in Virtual Environments

To fully utilize complex models and geometry, engineers must be able to wrap their mind around the information presented, which is often complex and difficult to visualize in the real world. Engineering prototypes can be large and complex and contain locations that are inaccessible. Analysis for these models can also be complex and difficult to extract data that is usable and understandable. Often, an analyst is needed to interpret the information gathered from a simulation. Creating an environment for design and analysis visualization and modification would improve the decision-making process. A synthetic three-dimensional environment can facilitate human capabilities for evaluation and decision making by accurately representing real-world experiences (Comparto, 2003). Engineering software tools such as computer-aided design (CAD) and computational fluid dynamics (CFD) packages create visualization environments that are capable of aiding designers and analysts in evaluating models.

Engineers often utilize high-fidelity models to investigate issues within a system that they may not otherwise understand. CAD and CFD packages aid in the visualization and analysis of these high-fidelity models, providing designers and analysts with insight into the system. Generally, designers use CAD to design a product to meet such demands as productivity, cost, appearance, and appeal. They use their knowledge of the market and consumers to create products they believe will meet these demands. Once an initial design is proposed, analysts examine the design using tools such as CFD or Finite Element Analysis

(FEA) to determine its quality. Upon examining the data, the analyst may have suggestions for design changes and must return the model to the designer to make the necessary changes. While CAD and CFD tools are collectively essential to the design process, the method by which they are used has its limits. Useful information can be extracted from each, but alterations to the design can be inefficient. An analyst may find useful information for a design change but must consult the designer for a new model in which to run the analysis. The designer may have additions or modifications to a design, but must wait for the analyst to complete the simulation before the model can be examined. This back-and-forth arrangement makes it difficult to explore possible alternatives in a timely manner. While no software is currently available that offers an environment where a design can be altered and analyzed, virtual engineering tools can create a comprehensive environment by coupling the CAD and CFD data together.

2.3 Interactive Virtual Environments

The design process has become more efficient over the past few decades due to new ways of thinking and integration of methods for collaborative thinking. Collaborative design can only streamline the process and improve the design process's efficiency. Bringing together several experts provides more information on possible solutions to a proposed design. However, problems can still arise within the design process due to lack of communication, designers not conveying thoughts clearly, delays between analysis, and bias on the part of an expert. While some of these issues may not be resolvable, others can be resolved through collaboration to create a comprehensive interactive virtual environment.

One goal of virtual engineering is to create a comprehensive decision-making environment that allows designers access to all aspects of the models through computer

interfaces. This environment must be a collaboration of experts from all fields to ensure that the questions under investigation are answered. Communication between experts is crucial in developing this synthetic environment to ensure that all aspects of design and analysis are covered. For example, in the proposed work, the designers are interested in redirecting air from a fan using baffles to attain proper air velocities and direction, but must rely on the CFD analyst for results of each baffle design change. To create a comprehensive tool that handles both the design and CFD analysis, clear communication between the designer and the analyst must be established. This communication should lead to a clear definition of the questions that the comprehensive tool within the virtual environment needs to answer.

The concept of creating a virtual environment for the purpose of design has been utilized in many areas of industry. For example, virtual environments have been used to help lay out floor plans for manufacturing buildings. Kesavadas et al. (1999) saw the benefit of this tool, noting that a well-laid-out manufacturing shop floor would help reduce idle time including transportation between machines, reduce bottlenecking, and improve the overall parts flow through the shop floor.

Xiao et al. (2004) discuss work done with a hydraulic mixing nozzle to optimize the nozzle exit/inlet ratio, which is known as the magnification. Genetic algorithms (GAs) were used to optimize the contour of the nozzle to create the maximum magnification factor. This project used model reduction to create CFD models capable of fast computational runtimes while maintaining a high level of accuracy. This is important in an optimization routine where hundreds or thousands of design iterations are done to find an optimal solution. In this study, the CFD model was reduced to contain about 22 times fewer cells while maintaining an error of less than one percent. This is an acceptable tradeoff when it comes to accuracy

and computational runtime considering the number of possible solutions that are evaluated in a short period of time.

Creating optimization routines coupled to CFD models has also been utilized in the design of coal pipes to help prevent what is called "coal roping." As the gas-solid mixture flows through the bend in the pipe, solid particulate separates out due to gravity and creates high particle density areas known as "coal ropes." A virtual environment was created to allow designer intervention as needed to design parameters using an interactive user interface. Huang et al. (2004) combined evolutionary algorithms (EAs) with a low-fidelity CFD model to shorten the design process for coal transport pipes while allowing designer interaction to help reduce the search space for the design parameters.

Cooking stoves used in third world countries use biomass as their main source of fuel. Initial work with these stoves increased fuel efficiency, decreased fuel use, and reduced particulate emissions by creating an evenly heated surface through baffle placement within the stove (McCorkle et al., 2003). This was done using an optimization routine known as graph based evolutionary algorithms (GBEAs). While this method provided several possible solutions, it does not account for economical or manufacturing constraints. Including designer knowledge and intervention in the design process provides access to the design variables that address the issues of economical and manufacturing constraints. Muth (2006) created a tool that allows designers to use their knowledge of stove design and thermodynamics to place baffles to redirect heated air within a stove. A user interface gives access to design variables to arrange baffles within the stove in a virtual environment. Using a virtual environment with an integrated CFD solver allows information to be passed

between the two environments, allowing results of different baffle arrangements to be seen in a timely manner.

Studies in bioenergy feedstocks show that some corn and wheat stover is of higher value as a biofeedstock than other residue. Kenney et al. (2005) applied the concepts of virtual engineering to the concept of selective harvest of this higher-value residue during grain harvest while still addressing concerns of soil sustainability and erosion control. Methods developed for the analysis of the selective harvesting combine include high-fidelity CFD models and particle image velocimetry (PIV) techniques to quantitatively and qualitatively characterize the cleaning shoe performance. A virtual environment was used to visualize these types of data, but further analysis of possible designs was desired. The separation of the flow within the selective harvesting combine was created by a single baffle. A baffle placement tool was developed that allowed designers to attach a baffle to a fixed point on the harvester at some rotation. This method allowed designers to test multiple baffle designs before building physical prototypes for testing. After field testing a baffle design created in the virtual environment, it was concluded that an acceptable design had been found.

Each of these projects relates to the presented work in some form, whether it is the creation of a virtual environment or model reduction for the purpose of improving computational analysis. The virtual environment creates a link that allows the designer to handle both the design and analysis tools. While the models presented in this project are more complicated than those of the previously defined work, the general goals remain the same: to create an integrated virtual environment with interactive capabilities and to reduce the computational analysis while maintaining an acceptable level of accuracy.

2.4 VE-Suite

As noted previously, one important goal of virtual engineering is to create an engineering decision-making environment that allows designers access to design and analysis tools through a single application. This means designers must be able to modify geometry in the same fashion as in CAD software while also being able to access simulation data as they would from analysis tools. Because analysis software such as CFD and FEA lacks the necessary tools for modeling and design, and design tools such as CAD lack the means to complete a thorough design analysis, a coupling of the two creates a power design tool. The Virtual Engineering Suite (VE-Suite) is one current virtual engineering tool capable of accomplishing such a task.

VE-Suite is an open-source software package developed by Dr. Mark Bryden and the Virtual Engineering Research Group at Iowa State University. Features of VE-Suite include platform independence, distributed computing, extensibility for component models, and comprehensive graphics capabilities including immersive environments. The main goal of VE-Suite is to enable users to incorporate component models and corresponding two- and three-dimensional graphical representations to create new plug-and-play components. (Bryden et al., 2004). Engineers can couple and utilize design and analysis tools in a single environment using the three main components of VE-Suite: VE-Xplorer, VE-Conductor, and VE-CE.

VE-Xplorer is the graphical representation of the model and allows interaction with the geometry and an extensive choice of data visualization within the virtual environment. This allows designers to fully analyze and interact with the model and data in real time. VE-Xplorer's visualization capabilities range from desktop computers to immersive virtual

environments such as a multi-wall system. These options are dependent upon preference and need. Typically, engineers can get by with visualizing data and geometry in a desktop environment, but at times it may be necessary to utilize a facility with multiple walls to fully experience the breadth of the model and data.

VE-Conductor is the graphical user interface (GUI) that allows interaction at a systems-based level as well as functionality for loading and manipulating geometry and data. Through VE-Conductor, users can perform basic program functionality such as opening and saving files and shutting down the software. The interface also provides access to geometrical and analytical information such as properties and general graphical requests.

VE-CE is the computational engine that handles the synchronization between VE-Suite and other outside analysis tools through a direct CORBA link. This allows information from the virtual environment to be passed into other analysis software for analysis and the results to be passed back into the virtual environment for visualization. VE-CE allows integration of many forms of analysis into the working environment and also allows distributed and cross-platform networking between computers.

VE-Open is a proposed open-source communication standard being developed at Iowa State University. VE-Open allows the VE-Suite components to be integrated and utilized more conveniently by providing the communication mechanism among the modules. VE-Open utilizes self-describing XML data structures to facilitate communication among the core VE-Suite engines. The VE-Open standard also serves as an API for integrating outside software.

2.4.1 VE-Suite Plugins

A general and flexible framework must be established to create an interactive environment for designers to utilize intuitively. This is accomplished through the VE-Suite framework. VE-Suite allows what is termed as plugins to be "plugged in" to the software framework to allow coupling of outside software with the virtual environment. Plugins are developed using the VE-Open specifications, which allow information to be passed between the virtual environment and outside software. To create an interactive environment within the VE-Suite framework, a plugin must be developed to coincide with each of the three VE-Suite engines. Therefore, a graphical user interface (GUI) plugin, a graphical plugin, and a computational unit plugin must be created to allow design and analysis within the virtual environment.

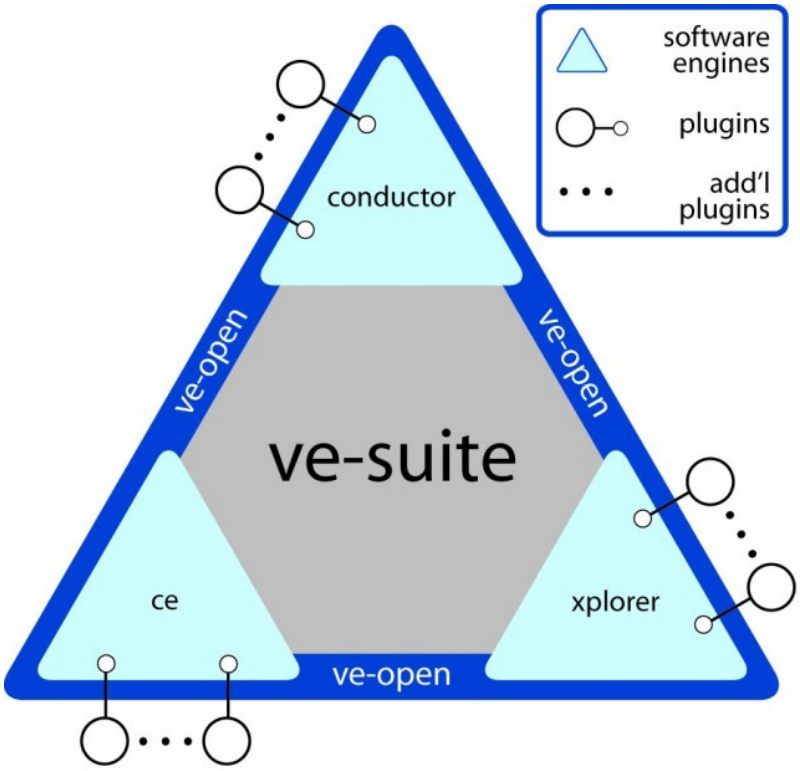

Figure 2. VE-Suite framework

The GUI plugin is accessed through VE-Conductor and gives designers control over design parameters. The design parameters are predetermined and address the questions that the designer would like to answer for a given model. The GUI plugin is created using an open-source cross-platform library called wxWidgets that provides functionality to create an intuitive and intelligent interface. An intelligent and intuitive GUI means the GUI controls must be aware of each other and update appropriately to any change to design parameters. The GUI should also allow the designer to move through the design process smoothly. This provides a simple yet robust interface for the designer.

The design process begins by opening VE-Suite and adding the module associated with the plugin to the design canvas. This loads any geometry or data that is associated with the module and allows the properties of the geometry or data to be accessed through the module. The next step is to use the design canvas from VE-Conductor to access the interactive design GUI plugin. The GUI allows designers to dynamically interact with the model through controls that modify the predefined design parameters.

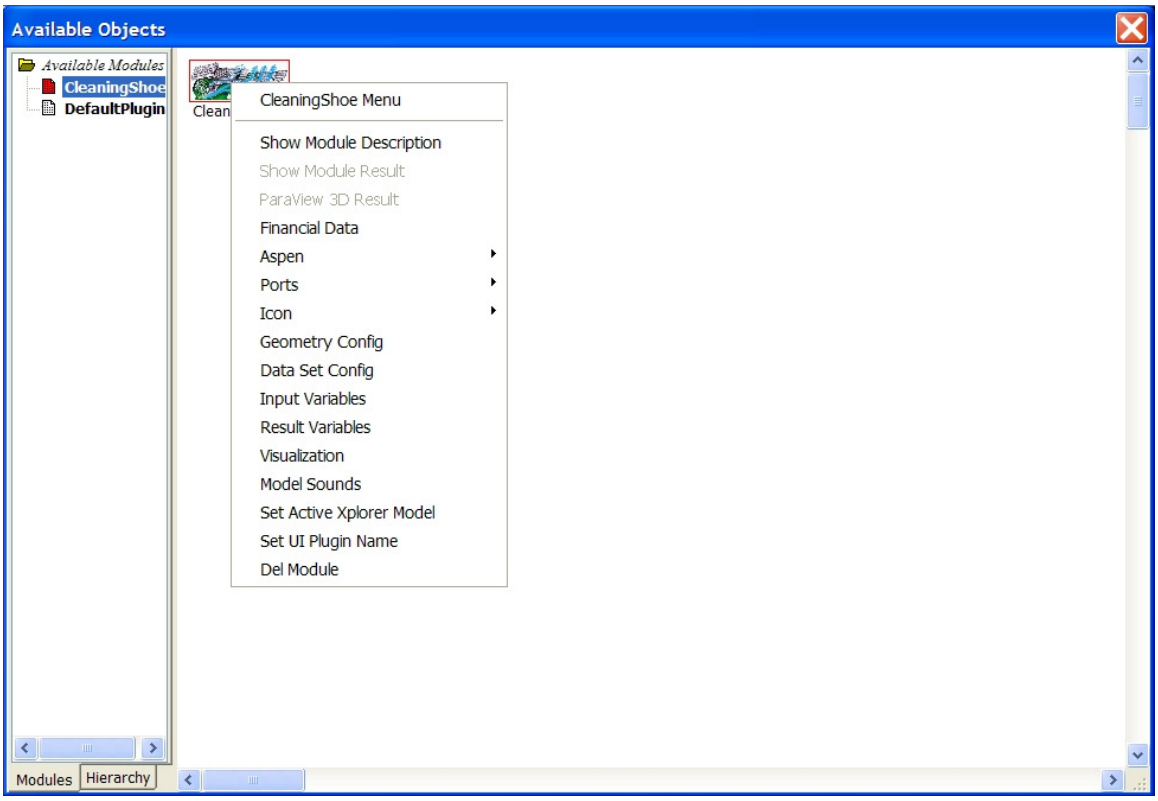

Figure 3. Module on design canvas with geometry and data options

The graphical plugin communicates with VE-Xplorer to update changes in the virtual environment relative to those done with the GUI plugin. The virtual environment in which the designer views the geometry and data must be adequate in that it should perform all the necessary tasks required by the designer to make a rational decision about a design. Through the XML data structure, the graphical plugin dynamically receives information about design variables that are being modified on the GUI plugin and visualizes the changes within the virtual space. The interaction between VE-Xplorer and the graphical plugin begins when the module is loaded onto the design canvas from VE-Conductor. Any geometry or data associated with the module is loaded into the virtual environment. Geometry that is not related to the system but that is helpful for the design process can also be introduced into the virtual environment. For example, in the case of the cleaning shoe graphical plugin, a

wireframe volume that represents the design domain where baffles can be placed as well as reference geometry is drawn in the virtual environment. The virtual environment also allows designers to find satisfactory designs through visual inspection of the baffle configuration before submitting the design for analysis. Once the analysis is complete, the designer can observe the results in the virtual space through the framework provided by VE-Suite. The visual representation of the results can help give direction for better decision making about baffle placement.

The computational unit plugin handles the integration of analysis tools by creating the proper avenues to deliver information in a way that is appropriate for outside software to recognize. It is a standalone executable developed and built to adhere to the VE-Open specifications. The computational unit plugin receives information from VE-CE and passes the information on to the proper outside application depending on the use of the computational unit. The computational unit delivers design variables or other forms of data to outside software either directly or indirectly through scripts for analysis. The executable is started through a command shell in which the proper environment paths have been specified.

2.4.2 StarCD™ Integration

The VE-Suite framework enables coupling design and analysis tools through the use of plugins. To create an interactive tool with design and analysis capabilities for the proposed project, it is necessary to integrate Star-CD™, a commercial CD-Adapco CFD software package, into the virtual environment. This is accomplished by coding within the computational unit system calls that allow access to the CFD model. Star-CD™ functionality can be accessed through command-line entries. Therefore, the scripting language is well defined and, if necessary, complete CFD models could be built through a series of line

commands. The robustness of the scripting language allows modifications to a CFD model without completely regenerating the model. This is important for creating an interactive model. The ability to modify portions of the model reduces the process time of restructuring the model for each design change. Therefore, it is important to have a well developed, converging model to allow an easy transition when small modifications to the model are desired.

2.5 Previous Interactive Tool Research

The work completed by Gent (2006) helped lay a foundation upon which an interactive tool could be built. This involved developing methods for creating reduced yet accurate CFD models. The CFD model that the initial work was completed on was an engine platform. The first step for creating a reduced model is identifying the critical area of the mesh. The critical area, or area of interest, is the area within the CFD model where changes will address the questions being asked. In this CFD model, the critical area was identified as the underhood compartment of the vehicle along the side of the engine. Grid studies were performed starting at areas that would have the least effect on the air within the engine compartment. This was determined to be the grid that represented the air around the outside of the harvesting machine. The grid was methodically removed until the CFD model of the harvester was reduced from the original model of eight million fluid cells to two million fluid cells. Upon conclusion of the grid reduction, the CFD model was reduced to contain only the engine compartment of the harvester. The removal of cells had a significant impact on reducing the computational runtime of the model. Removing certain portions of grid further improved the transformation of the model into an interactive model. Grids containing complex boundary conditions were removed and replaced with simpler boundary conditions,

which had a major impact on the model.

The CFD model had been reduced to contain mainly the engine compartment with the radiator fan, which is the main source of air movement, missing. Because the remaining CFD model was downstream of the original radiator fan boundary condition, other boundary conditions had to be applied to the surface of the grid. This meant that the air properties along the outside of the reduced model had to be found and applied to maintain the appropriate level of accuracy. Gent and McCorkle (2006) developed a method that extracts the air flow properties from the original model at the interface that is represented by the reduced model. Through the utilization of scripts and functionality in Star-CD™, the air flow properties are converted to inlet boundary conditions. Scripts are then used to transpose the information from the shells into inlet boundary conditions for the reduced model. The scripts open the model in Star-CD™ and apply the inlet boundary conditions as if an analyst were applying each boundary condition individually and defining the properties for each one. This work involving grid reduction and boundary condition manipulation improved the computational runtime from three days to just over three hours while maintaining an acceptable level of accuracy.

Although three hours is a significant improvement over three days, there was still a need to further reduce the model's computational footprint to try to attain a simulation runtime of less than hour. Therefore, the methods developed were continued as other areas in the model were examined for importance. Continuing this work proved to be difficult because the model had already been greatly reduced to include only the entire engine compartment. This means that a majority of the remaining grid most likely has some impact on the air flow that occurs within the area of interest. From here, careful grid selection is

needed when considering which cells should be deleted from the model.

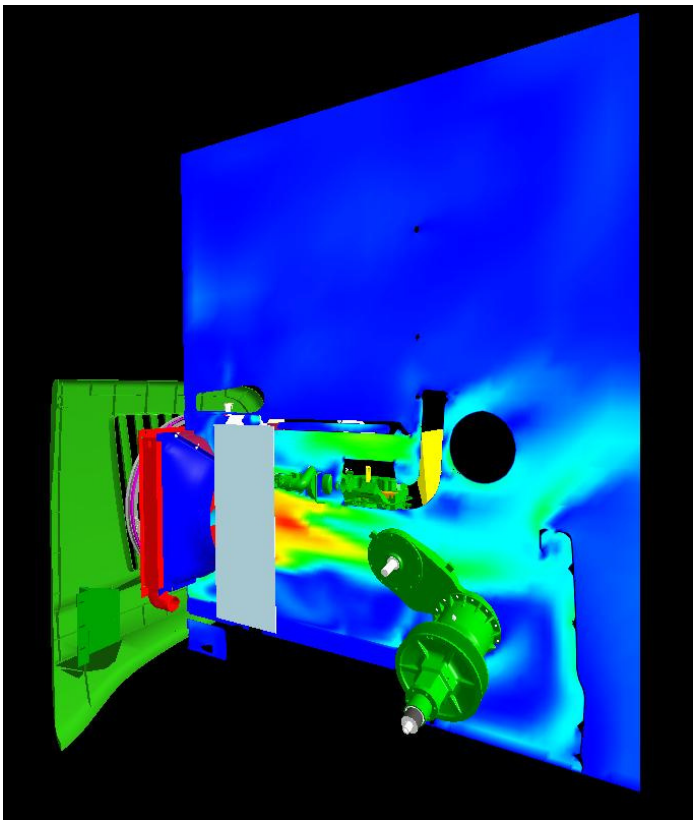

Figure 4. Baffle in engine platform system to redirect air along engine

The previous work addressed one aspect of creating an interactive design tool. The main goal of this work was to create an interactive baffle placement tool that integrates design and analysis into a single environment. Initially, this tool was created specifically for the harvester engine platform model but was quickly found to be useful in other areas of the machine as well. The environment and interface in which the designer interacts with the CFD model still needed to be developed. Creating an environment that meets all of a designer's needs is important to ensure that the designer can make the appropriate decisions for the questions that are being asked. This includes having an intuitive and intelligent interface that allows the designer to navigate easily through the design process. Secondly, the virtual space must allow easy exploration of geometry and data and have the capability of

interacting with the both. Lastly, the computational engine must be proficient in handling the information being passed between the virtual environment and the analysis tools. The design and analysis tools must be able to work seamlessly and robustly to avoid complications during the design process.

The work completed in this thesis involves two models: an engine platform and cleaning shoe of a harvesting machine. While the technical aspects of these projects were quite different, the overall goals for both projects were generally the same; that is, how to redirect air to create proper air flow within the respective area using baffles. Therefore, the finished product of the interactive tools for both projects is similar. One of the goals of this work was not only to create an interactive baffle placement tool, but to create a tool that was generic and could be easily modified for different locations within a system or for completely different systems. Initially, this tool was created specifically for the harvester engine platform model but quickly found usefulness in other areas of the machine. As the work scope evolved from the engine platform to the cleaning shoe, functionality was added to make the tool more generic for easy transition between models. Over the two years this tool has been under development, it has evolved to envelop new capabilities brought about by enhancement requests and new capabilities with the virtual environment provided by the VE-Suite framework. The VE-Suite framework provides a nice coupling between itself and the plugins to aid in the creation of a robust, interactive design tool.

CHAPTER 3: INTERACTIVE TOOL CASE STUDIES

Two models were examined for this work, both with the purpose of studying air flow systems and how to direct air to or away from areas within a combine. One way to redirect air is through the utilization of a baffle. Because of the complexity of the equipment and space requirements within the machinery, only certain areas allow the placement of baffles. Creating a method to design and analyze baffle arrangements in a virtual environment would have benefits on many levels. The current process of designing and testing baffles is cumbersome and expensive. Field engineers with velocity-measuring equipment place makeshift baffles into the farm machinery in areas they believe would improve the air flow. This trial-and-error method has proved to be time-consuming and inefficient. The goal is to create a way to design baffle arrangements and analyze them in a timely fashion within a virtual environment. In the work presented in this thesis, two air flow systems were examined for interactive baffle placement: a harvester engine platform and a harvester cleaning shoe. Both systems utilize proper baffle placement for air redirection, but for different purposes. While the systems studied in this research relate to farm machinery, the interactive baffle placement tool could be utilized in any air flow system where air flow management is a concern.

3.1 Harvester Engine Compartment

Harvesting machines, like other agricultural equipment, must be able to withstand harsh conditions while operating in the field. These conditions include dust and debris that may be circulating in the air and that can enter the engine compartment and settle on various engine components. The radiator fan used to cool the engine can aid in preventing debris accumulation by providing high-velocity air to these components in the engine compartment.

However, the engine compartment of a combine is sizable and the air moved by the fan quickly disperses. Thus, the air velocity rapidly dissipates as it moves through the engine compartment. As the velocity decreases, the debris begins to settle out of the air and onto the engine and other parts within the compartment leading to several unfavorable effects, including decreased machine life and the potential for fire created by heat generated from the engine and other parts within the compartment. To reduce these effects, high air velocity needs to be maintained at the collection-prone regions of the engine compartment. A current method to increase air flow and reduce the settling of particulate is to place baffles within the engine compartment to redirect air to the areas where settling occurs.

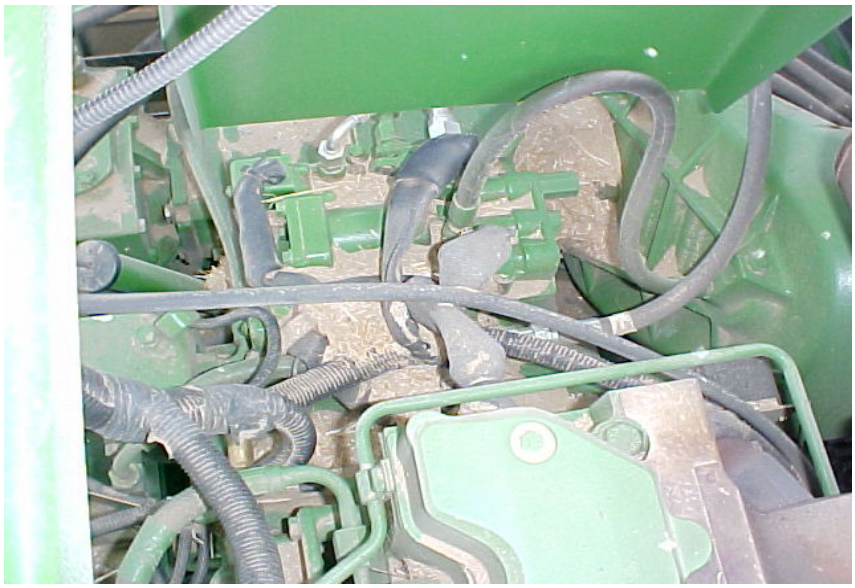

Figure 5. Harvester engine compartment with debris accumulation

Until recently, there has not been an efficient method to test the effectiveness of the baffle placement within the engine compartment. The introduction of computational fluid dynamics (CFD) into the design process allowed analysis to be conducted for each baffle design. However, this has also proved to be time-consuming due to the large, complex models and long computational analysis. This project's goal was to demonstrate the

capability of creating an interactive baffle placement tool that could be used in future applications that would need deflectors to redirect air within a given space. This case study set up the framework for the interactive baffle placement tool that would allow simple modifications to transfer the tool between applications.

Finding an ideal baffle configuration that maintained acceptable air flow through the engine platform was the question that engineers wanted to address for this application. Creating the interactive CFD model meant that an area of interest must be identified within the full-scale CFD model. An area alongside the engine was chosen due to the settling that was occurring as well as the openness of that location in the engine compartment. This provided an ideal location to apply CFD model reduction methods and to test the interactive baffle placement tool capabilities.

3.2 Harvester Cleaning Shoe

The cleaning shoe within a harvesting machine plays the role of separating the grain from chaff and other particulate. The cleaning shoe is fed through a set of augers that move the harvested material to the front of the cleaning shoe where the grain and particulate is met with a rush of air from a fan. The fan creates what resembles a fluidized bed, allowing the grain to fall out of the mixture through a series of sieves and chaffers while blowing the remaining particulate out the back of the machine.

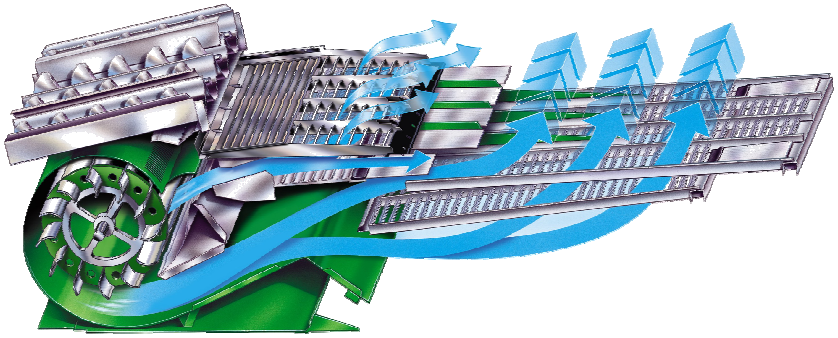

Figure 6. Harvester cleaning shoe

Proper air movement within the harvester cleaning shoe is crucial in the separation process. Several factors affect how well the separation occurs while the grain and particulate move across the separators including fan speed, deflectors, and sieve and chaffer openings. A fan speed that is too low can lead to particulate not being properly removed from the grain while a high fan speed could blow the grain out the back of the machine. Having the correct opening size for the sieves and chaffers affects the amount of air utilized to push particulate out the back and how the grain settles out of the mixture. Deflectors can be placed in the cleaning shoe to direct air to locations that may not have sufficient air movement, particularly under the sieves and chaffers. From field test and CFD analysis, it is known that for the current cleaning shoe design, neither air flow from side to side nor front to back meets the standards set by the designers. The air distribution from side to side is not consistent and the air flow from front to back does not create an even fluidized bed across the sieve and chaffers. Keeping an even distribution of air across the system is important to prevent plugging or overloading of harvested material at any location on the separators. Baffles can be placed throughout the cleaning shoe to divert air to create a more stable and predictable fluidized bed. The issue of creating an even flow from side to side deals with getting consistent air flow from the fan, which is currently being addressed with a full-scale model

by other analysts. A good fit for the interactive baffle placement tool is to study the air flow from the fan exits to the back of the cleaning shoe. Focusing on the air flow from the front to the back of the cleaning shoe allows designers to create an adequate system that will work with the resolved fan system.

With simple modifications to the interactive baffle placement tool framework developed for the harvester engine platform, baffles could be placed in designer-specified locations within the cleaning shoe system. The interactive baffle placement tool was further developed in this work to include more features for the designer. The goal of this work was to create an interactive baffle placement tool that could be utilized within the current design process of the cleaning shoe.

CHAPTER 4: CREATING AN INTERACTIVE MODEL

Creating a computational fluid dynamics (CFD) model that has the ability to be interactive means having a well-defined, accurate model with the capability to be modified and perform simulations in a timely manner. In this study, two CFD models were examined and different methods were studied to make them interactive. Both cases dealt with large, complex models that had computational runtimes of several days. An interactive model must be able to run a simulation within a timeframe that is acceptable to the designer. Therefore, several approaches were considered to reduce the computational runtime of these models, including cell reduction, cell coarsening, and modifying boundary conditions. To make the proper decisions regarding model reduction, analysts must be aware of the areas that designers are interested in studying and how other parts of the model affect those locations. If those questions can be answered, a model can be greatly reduced through the methods of model reduction.

The first grid study consisted of an outdoor environment containing an entire harvester. The full model originally contained over eight million fluid cells. The first step in creating an interactive CFD model is determining the questions that needed to be answered. In this case, the solution would be to help improve air flow along the side of the engine by redirecting air being moved by the radiator fan. The model contained boundary conditions representing wind as well as a moving boundary condition to represent the radiator fan.

The second case study also involved a harvesting machine; more specifically, the cleaning shoe system. Designers need to address several important issues associated with air flow in this model. The focus for this model was to get the proper air flow from the fan to the back of the cleaning shoe. The full scale model of this cleaning shoe contained over

sixteen million fluid cells. The boundary conditions for this model also consisted of a moving mesh to represent the fan. It is obvious that in both cases, creating an interactive model would require that these cell counts be greatly reduced and boundary conditions altered.

4.1 Cell Reduction

Cell reduction is the most effective method for reducing the computational runtime of a model, but attention to detail is also necessary because removing the wrong section of grid can have a huge impact on the model's accuracy. Analysts with knowledge of the system and how system components affect air flow should be responsible for removing grids from the model. Grids should gradually be removed from the model with analysis being completed after each iteration to ensure that the model maintains an acceptable level of accuracy. Two models were examined in the first case study. Different questions had to be addressed for each, so the method in which cell reduction was approached was different for each model.

4.1.1 Cell Reduction for First Case Study

The decision to further investigate methods to reduce the computational runtime on the harvester engine platform model to make it more interactive proved to be successful for two reasons. First, using the methods established for CFD model reduction enabled the model to be reduced from two million fluid cells to just over eight hundred thousand cells. The drastic change in the amount of grid is due in part to the fact that the remaining grid was surrounding the area of interest. Often, the grid in CFD models gets finer the closer the grid gets to the area of interest to provide greater accuracy at and around that location. The grid on the opposite side of the engine from the area of interest was the first to be examined for

removal. The air movement on the opposite side of the engine had little impact on the area of interest and allowed deletion of cells almost up to the engine. It was found that reducing the grid from two million cells to eight hundred thousand cells cut the computational runtime in half, from three hours to an hour and a half.

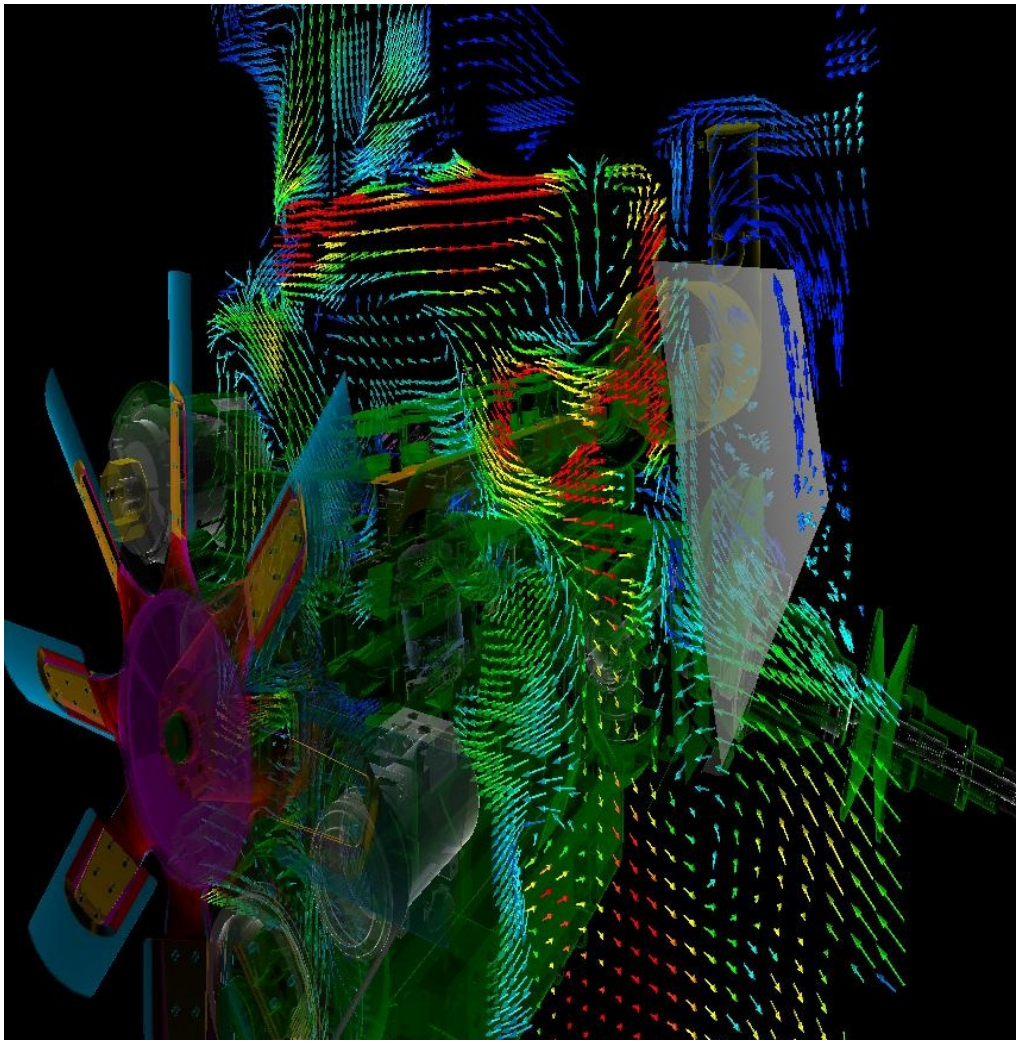

Figure 7. Engine platform with baffle and velocity magnitude vector plane

Star-CD™ provides functionality to map flow conditions and properties from one model to another. The command *PMAP* takes the post file, the data file created by Star-CD™ at the end of a simulation that contains all the simulation information, and maps those

properties to the new model, which provides a reasonable initial field guess for model restarts (Star-CD™ User Guide). Therefore, cells that are not necessarily affected by the design change do not take much computation to resolve. This method is only applicable if the model that the data is being mapped to has the same volume as the model in which the post file was taken. Adding this functionality to the simulation further reduced the computational runtime to 45 minutes on a single processor.

4.1.2 Cell Reduction for Second Case Study

The full-scale CFD model associated with the cleaning shoe system contained over 16 million fluid cells and had a computational runtime of nearly one week. This model needed to be drastically reduced in order to consider using the CFD model with an interactive baffle placement tool. Once again, designers had to ask the questions they wanted answers to by using the interactive baffle placement tool. There are several issues related to the air flow throughout the cleaning shoe. Designers and analysts agreed to focus on utilizing the interactive baffle placement tool to study air flow moving from the front to back of the cleaning shoe starting at the fan outlets. Unlike the underhood engine compartment of the first case study, the cleaning shoe system generally has a symmetrical design from side to side. This presents a great opportunity for grid reduction. Since the question to be answered addresses air flow from the front to the back of the cleaning shoe, a two-dimensional cross-section of the model can be analyzed. The model was further reduced by excluding the fan, which, like the engine model, simplified the inlet boundary conditions. Applying these techniques reduced the model from 16 million fluid cells down to just over one hundred thousand.

4.2 Boundary Conditions

The type and number of boundary conditions in a CFD model can play a significant role in its accuracy, convergence, and simulation time. Maintaining proper boundary conditions is necessary when reducing a model. This can be done using several methods. The CFD solver may come with functionality to allow boundary conditions that are suitable for a reduced model, but sometimes the analyst must devise methods for accounting for boundary conditions that are lost from the reduction process. Each case study utilizes one of the methods mentioned above.

4.2.1 Boundary Conditions for First Case Study

In the case study involving the air flow around the harvester engine platform, the CFD model originally had boundary conditions to represent a head wind and the movement of the radiator fan. Earlier work determined that the head wind did not significantly affect air movement within the engine compartment, which means that the radiator fan was the main source for moving air through the engine compartment. To maintain a high level of accuracy, it was necessary to model the boundary conditions of the radiator fan using an external subroutine in Star-CD^{TM} to simulate the fan rotating at a specified number of revolutions per minute. This allowed the model to be extremely accurate, but extended the model's computational runtime.

When making decisions about creating an interactive model, one must consider the purpose of this tool. The goal is to gain multiple design alternatives that help give direction to a design while maintaining an established level of accuracy. Thus, the most accurate model is not necessarily the best model when it comes to creating a model that has interactive capabilities. Using this mentality, the boundary conditions can become less accurate and

more tailored toward creating a quicker converging model. To create this interactive model, the subroutine representing the moving fan must be removed and replaced with stationary boundary conditions.

Since the fan provided the major source of air movement within the engine compartment, boundary conditions needed to be created around the entire reduced model to simulate the fan's influence on the air within the engine compartment. Creating these boundary conditions requires the use of techniques developed in earlier research. Utilizing those methods to create boundary conditions could create thousands of inlet boundary conditions for the reduced model. Even though the model has been greatly reduced, the large number of inlet conditions can actually increase the computational runtime. Applying inlet conditions around the entire model would be incorrect because areas of the model have outflow away from the engine compartment. As expected, the inlets with the greatest velocities are near the location of the original radiator fan. It can be observed that some of the inlet boundary conditions around the entire surface of the model had outflow velocities especially toward the top and back of the model, opposite the radiator fan. To help the model converge more quickly, the inlet conditions with large outflow velocities were converted to outlet boundary conditions.

One issue surrounding the application of boundary conditions near the area of interest is the effect that the baffle will have on the air flow at those conditions. Areas that may have had outflow and were converted to outlet boundary conditions may not be a correct assumption after a baffle is placed into the system. Similarly, inlet boundary conditions such as velocity magnitude and direction could change due to the addition of the baffle into the air flow system. When using the methods developed from earlier research, it must be assumed

that there is one-way propagation through the CFD model, meaning that the effects of the baffle do not affect the current boundary conditions and do not require the application of new boundary conditions. Taking into account the baffle for changing boundary conditions would require an iterative process where new boundary conditions would have to be applied after the simulation runs for a few iterations, making it difficult to make the model interactive.

4.2.2 Boundary Conditions for Second Case Study

The second case study involving the cleaning shoe was approached differently due to the symmetrical features of the design. The symmetry of the model allowed functionality within Star-CD™ to be utilized for boundary condition application. Originally, the full-scale cleaning shoe model used a rotating boundary condition similar to the engine platform model to represent the fan. Using some of the same methodology applied to the engine platform model allowed the fan to be removed from the CFD model and replaced with inlet boundary conditions that represent the fan outlets. Experimental data from field tests were used to obtain the inlet boundary conditions to represent the fan outlets.

As mentioned earlier, the cleaning shoe CFD model is being analyzed as a two-dimensional problem due to the cleaning shoe's symmetry and the questions that are being addressed at this time in the design process. Star-CD™ provides boundary conditions called symmetry planes that were placed on both sides of the model to represent the symmetrical behavior of the model. Applying symmetry plane boundaries to each side of a two-dimensional CFD model signifies the plane of geometrical and symmetrical flow. Inputs for this type of boundary condition are not necessary. By applying these boundary conditions, it is assumed that the normal component of the velocity is zero and that the normal gradient of

all other variables is also zero. (Star-CD™ User Guide)

4.3 Coarsening Cells

Another method of reducing the amount of grid in a CFD model is through the coarsening of grid. This method was examined during the grid study involving the engine platform. Since this was a large, complex, three-dimensional model, coarsening the grid initially seemed like a promising method for cell reduction, but issues arose while trying to coarsen the model. Earlier work had reduced the CFD model to a point where nearly all areas of the grid were important to maintain an acceptable level of accuracy. The grid around the area of interest was refined for the purpose of accuracy and coarsening an entire section of fine grid would lead to a loss of accuracy within this location in the model. Other areas were examined for coarsening, but the efforts in these areas proved to be futile. Coarsening these areas led to only a small percentage of cell reduction while at the same time being extremely time-consuming. Therefore, coarsening fluid cells to reduce the cell count was deemed impractical. This method was not examined for the second case study involving the cleaning shoe. The symmetric nature of the design and the questions that needed to be answered allow the use of a two-dimensional model with the complex boundary conditions removed. These features alone create a model that is adequate to use with the interactive baffle placement tool. The dimensionality allows the model to contain refined grid in areas in which designers are concerned about the air flow while still maintaining a relatively low cell count.

4.4 Designating the Area of Interest

Choosing and creating an area of interest is important in determining the overall outcome of the design and analysis. Designers and analysts should collaborate at the

beginning of the design process to discuss areas within the system that need to be addressed. At this early stage, clear communication between the designers and analysts is important. The designers and analysts must have a good understanding of the system and how changes within the design could possibly affect the air flow. This knowledge may come from previously recorded data, field experience, and earlier designs. This allows the designers and analysts to decide the questions that the interactive CFD model should address. Their understanding of the system and the questions the interactive CFD model should answer ensures that the analyst creates a CFD model that allows designers to design within a designated area of interest.

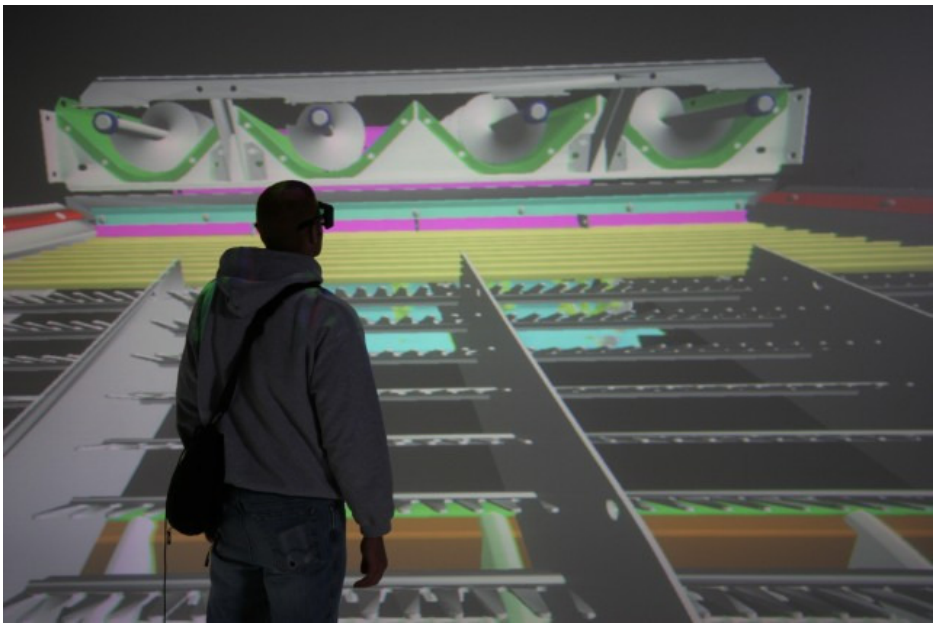

Figure 8. View of cleaning shoe in a 4 wall immersive environment

To create a CFD model with interactive capabilities, a simulation of the proposed full-scale CFD model is run to obtain initial results for system. The results from the full-scale CFD model should be validated with field tests or other data to guarantee that the

analyst is working with an acceptable model. Once the results of the CFD model are confirmed, the designers and analysts can shift their focus to examining flaws in the system. Upon reviewing the model, they can make suggestions for possible design improvement locations, which then become the areas of interest. Analysts can then reduce the model using the methods mentioned earlier to improve computational analysis time. The CFD model should be reduced to concentrate on a specific location or locations within the system, removing any areas that have little to no impact on the designated area or areas. Upon completion of the reduced CFD model, analysts can shift their focus to the preparation of the area of interest. This area is the agreed-upon location that the designers and analysts believe can positively impact the system and addresses the current issues with the system. Using the Generic Math Template Library (GMTL) functionality within the GUI plugin requires the area of interest within the CFD model to be a rectangular volume. This limits the functionality of the interactive baffle design tool, but simplifies collision detection between the area of interest and the designed baffles. It also simplifies CFD model preparation for the area of interest. This area needs to be prepared in such a way that allows for repeated remeshing without incident. This ensures that once an interactive CFD model is created, the designer will not have to return to the analyst for assistance with the model or for interpretation of the data.

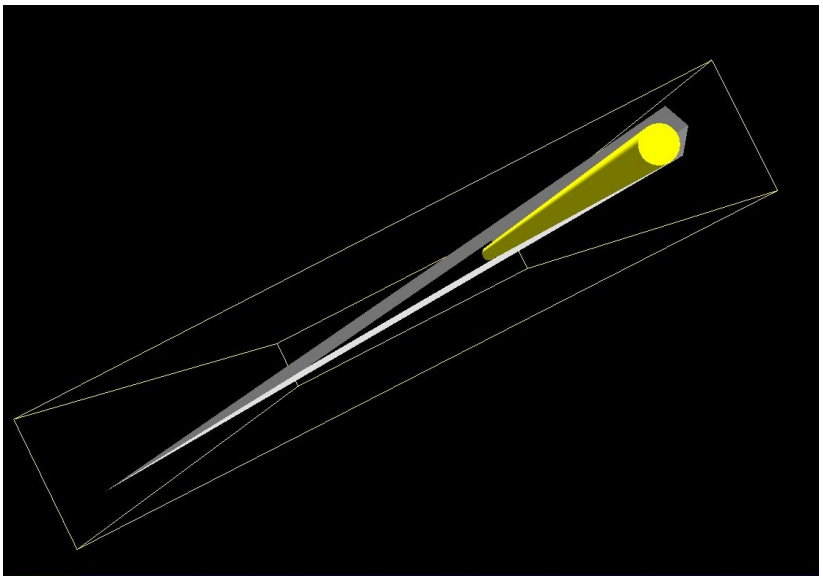

Figure 9a. Area of interest as seen in VE-Suite

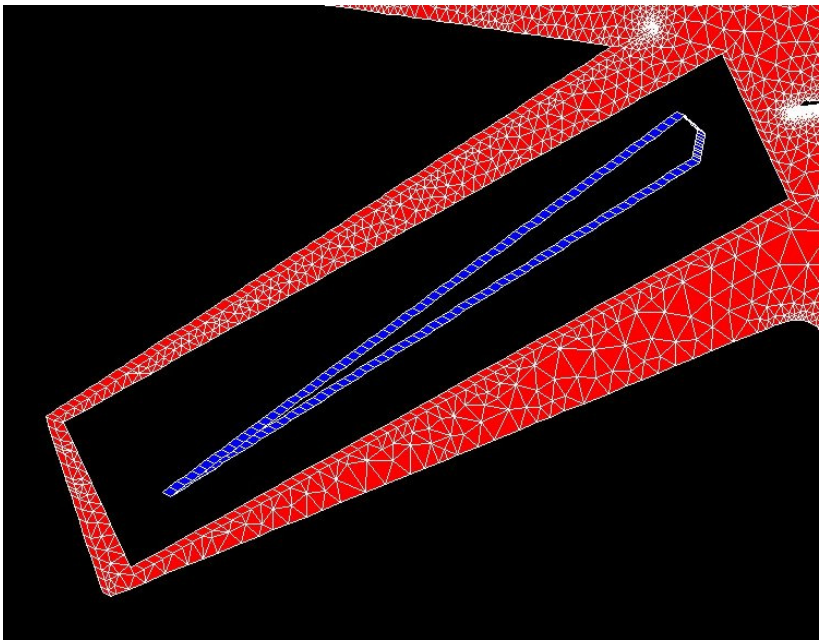

Figure 9b. Area of interest in Star-CD™ with baffles before remeshing

4.5 Creating Star-CD™ Scripts

With these designated areas of interest identified, corresponding areas within the CFD model can be modified to utilize the interactive baffle placement tool. The interactive CFD model must be structured properly such that a generalized script of commands can open the

model and make the necessary modifications over and over without damaging the integrity of the model. This is accomplished by utilizing database functionality within Star-CD™. By using databases, the base CFD model can always be accessed and saved into another database once modifications are applied. This ensures that with each design iteration, the quality of the base CFD model is maintained. This is important because it ensures that a designer will not have to consult a CFD analyst for assistance with fixing the model because a problem with the CFD model cannot occur. This keeps the CFD model and simulation behind the scenes and only allows simulation results to be exposed to the designer.

The first step in creating an interactive model is to create a well developed, converged CFD model. From this model, a base interactive CFD model that allows design manipulation to the area of interest should be established. The base interactive model consists of the reduced mesh except for the designated area for design. Through using the databases provided by Star-CD™, this design area can be remeshed with baffles defined by the designer and saved into new databases. Therefore, only a small portion of the overall mesh actually gets remeshed. This greatly helps reduce the process time of each design iteration. The new mesh is then coupled to the base mesh to create a new, complete CFD model. This process is facilitated by the creation of scripts containing the necessary Star-CD™ commands. Throughout the computational unit, system calls run these scripts, resulting in a new CFD model. Details of these scripts will be discussed later as different scripts were created to handle different design formations, giving the designer more freedom. Scripts are also created for post-processing. The first script opens Star-CD™ to save out files containing air flow data. Another script uses these files to translate the data from Star-CD™

to a format that is recognizable by VE-Suite. Therefore, the analysis and data translation occur behind the scenes, resulting in a dataset that is viewable within the virtual environment.

The scripts used to create the new CFD models are generally the same and can be easily transferred from one application to another. Minor modifications are needed within the scripts to ensure that the coordinates in the CFD model correspond to those used in the interactive baffle placement tool. Since the scripts consist of strictly Star-CD™ commands, the analyst should be the one to modify them for new applications and make certain that they perform their purpose at the correct location over and over without incident.

4.6 Creating an Interactive Environment

Creating an interactive environment requires the user to have the necessary tools to answer the questions addressed toward the issues of the current design. To accommodate the designer, plugins were created to interact with each of the components of VE-Suite framework. The framework of VE-Suite allows plugins to interact and use functionality within the software as well as with outside software to aid in the design and decision-making process. Each of these components allows access to different aspects of the interactive tool, either directly or indirectly. The user can directly interact with the GUI and virtual environment to manipulate geometry and data while CFD computations for baffle designs are indirectly accessed through the computational unit plugin. This provides a simple environment for users to design and complete analysis.

4.6.1 Interactive GUI

The interactive design GUI is made available by double-clicking the module on the design canvas in VE-Conductor. The GUI is laid out to help the designer step through the design of the baffle systematically. Designers can design up to four baffles in the designated

design domain and have control over the position, orientation, and size of the baffles. When the GUI is first brought up, the baffle controls are inactive to avoid issues with baffles being created outside the model. To make the controls active, a baffle plane must first be selected. Selection of a plane makes the baffle appear in the virtual environment at a preset dimension and starting location. The designer can then use the slider control or the spinner control to modify the baffle's length and width. The range of values for the length and width is determined by the plane that the baffle is created in. This ensures that the designer cannot create a baffle that is outside the domain of the design space. The designer can then change the starting location of the baffle if desired. The starting location is represented by a corner of the baffle, generally the corner that is initially in the corner of the design space. The coordinates of the baffle's corners are recognized within the tool, ensuring that the baffle does not intercept or extend the design space. Another option is to apply rotation to the baffle. Designers can rotate a baffle around the three orthogonal planes to the point where the baffle intersects the sides of the domain.

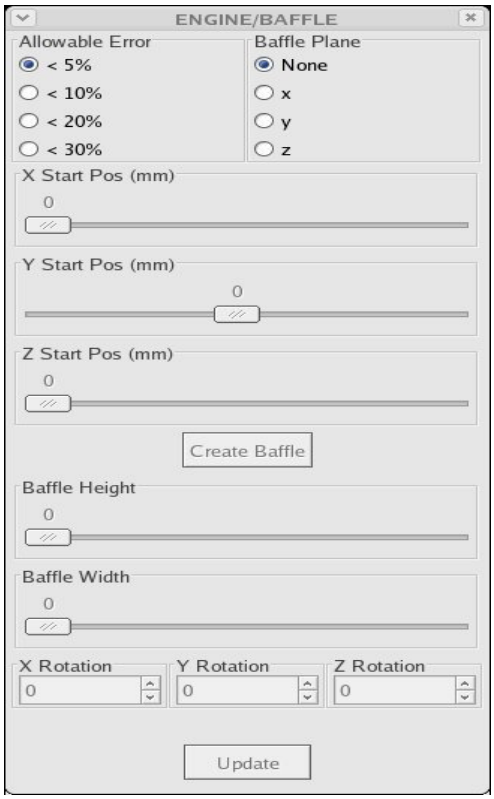

Figure 10a. Interactive baffle placement GUI for the engine platform application

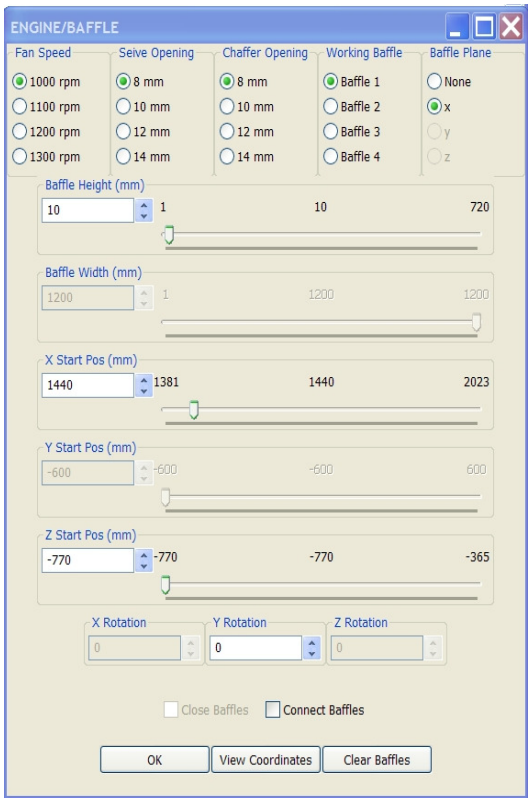

Figure 10b. Interactive baffle placement GUI for the cleaning shoe application

While all the features on the GUI are desirable for a three-dimensional model to allow full movement of the baffle within the design domain, a two-dimensional representation of a model can simplify the GUI. Removing a dimension from the model reduces the number of design parameters that must be identified to fully define the baffle. These design parameters, which are not applicable for a two-dimensional model, were made inactive on the GUI. For example, the two-dimensional cleaning shoe model no longer has depth in the y-direction, so the radio box containing the plane direction options could be simplified to just one plane. Also, because of the loss of depth, the width parameter along with the starting y-location parameter was no longer applicable and can be made inactive along with the spinner controls relating to rotation in the x-direction and z-direction.

As mentioned earlier, the GUI should be intelligent to ease the use of the GUI and allow the designer more freedom with design parameters. This intelligence was incorporated into the GUI so that the GUI controls would respond to changes by other controls. This can be displayed in many forms. Generic Math Template Library (GMTL) was used to create and manage events that affect the area of interest. GMTL provides functionality to create a bounding box and test for points within the volume. This provided an easy method for testing the corners of the baffles after each modification to ensure that they were within the design domain. The utilization of GMTL also enabled instantaneous adjustment of baffle parameter ranges. For example, when the plane direction is chosen, the range of the controls dealing with length, width, and starting location all adjust appropriately to update the design space domain. The same is true for length and width when the starting location is modified. The ranges of the length and width adjust automatically when the starting location is modified to ensure that the baffle does not leave the designated domain. If the baffle's length or width already intersects the bounding domain and altering the starting location would require the baffle to shrink, the controls automatically update on the GUI to reflect the baffle being shrunk to allow for that starting location. A slider control and spinner control are provided for the height, width, and starting locations of the baffle. If a slider control is used to adjust a baffle parameter, the associated spinner control will adjust accordingly and vice versa. Because of the functionality within wxWidgets, the sliders can only return integer values while the spinner controls have the capability to return double values. Therefore, all information passed from the GUI to the virtual environment is taken from the spinner controls any time a parameter is changed.

Another feature that demonstrates the GUI's intelligence is the connection box

option. Once the first baffle is created, the Connect Baffles box can be checked and the next baffle added to the design will be attached to the end of the previous baffle. This allows baffles to be connected in a series to simulate rough curves. When two baffles have been created and a third is desired, the designer has the option to close the baffles, creating a loop. This connects the baffle to the end of the previous baffle and to the front of the first baffle. These features create an intelligent design tool and aid the designer through the design process.

4.6.2 Interactive Virtual Environment

The virtual environment provides a visual component to the interactive design tool. It allows the designer to visually inspect baffle configurations and see how they fit within the system. Baffle geometry is dynamically updated in the environment, reflecting changes to design parameters within the GUI. Design and modification of the baffles can be seen within the virtual space provided by VE-Suite, allowing the designer to examine the baffle configuration before putting the design through the analysis process. Once the simulation is complete, the system can be visually analyzed through functionality within VE-Suite, allowing designers to make rational decisions about the further exploration of each design.

4.6.3 Computational Unit

The computational unit handles the passing of information from the VE-Suite to outside software. The computational unit is a stand-alone application that can be started by opening a command shell and setting all the necessary environment paths needed to properly run the executable. VE-Suite allows the option of opening a shell, which automatically sets all the environment paths. For simplicity, the computational engine is started using a script that contains a single line command that connects the application through CORBA to VE-

Suite. Once the executable is running, it can receive information from VE-CE regarding baffle parameters or any other data parameters that are needed for the executable to complete the required tasks. Each time a design is submitted for analysis, the executable handles a series of events that must occur in the correct order for the tool to work successfully. The first event is to delete existing cell and vertex files from the working directory. These are the files that are read into Star-CD™ that represent the baffles that were designed using the GUI plug-in. Before the new cell and vertex files can be created, the old ones must be deleted to remove any chance of the improper files being read into the new model. The new files that are created with each design submission have the same name as the previous baffle design and any following baffle designs. Therefore, it is necessary that the cell and vertex files from the previous simulation be deleted to prevent the possibility of disrupting the current analysis. If the designer would like to keep the previous baffle configuration cell and vertex files, they must be moved into a different directory.

The next step is to create the cell and vertex files that create the baffles in the CFD model. Through the XML data structure, the information about the baffle parameters is received and input into the computational unit executable. These parameters include the plane direction, length, width, starting vertex location in Cartesian coordinates, and any orthogonal rotation. The executable takes the baffle parameters and creates cell and vertex files using GMTL's matrix functionality to handle transformations and rotations. This ensures that the baffle corners match the coordinates of the baffle designed in the virtual environment and within the Star-CD™ model. The size of the individual baffle cells are determined by the length and width of the designed baffle. The larger the baffle created in the virtual environment, the more baffle cells that are created. This ensures that the grid

created around the baffles in the CFD model is of an effective size and maintains a high level of accuracy around the baffles. Since the cell and vertex file are being generated by code rather than saved out of a Star-CD™ model, the proper format for the cell and vertex files had to be hard-coded into the executable in order to allow Star-CD™ to read in the files correctly.

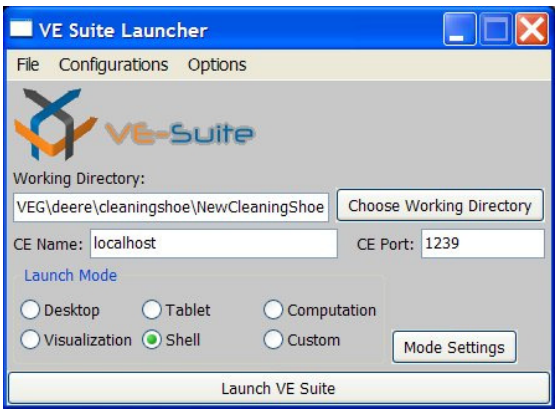

Figure 11. VE-Suite launcher to start VE-Suite and open shell

Once the cell and vertex files are created, they can be read into the CFD model. A structure must be made within Star-CD™ to allow modifications to the model for a new design. Star-CD™ provides the functionality of databases, which allows sub-models to be stored within a database file. Setting up several sub-models allows the baffle placement and new model creation to occur seamlessly. Models can be completely created and modified within Star-CD™ through command line entries, which makes Star-CD™ accessible through scripts containing a series of commands. After the cell and vertex files have been created for the designed baffles, the computational unit calls a script that opens Star-CD™ and begins the process of creating the new model. First, the baffles are read into an empty database. A database containing a sub-model of shells representing the outline of the area of interest is then added to the database with the baffles, and the complete cell set is outputted to another

database. Fluid cells are generated within this database to create the mesh for the area of interest. This database is then coupled with another sub-model consisting of the reduced base model except the area of interest. Boundary conditions along with analysis parameters are applied, completing the model and making it ready for analysis. The script then saves and closes the model and begins the analysis.

The scripts used to create new CFD models that reflect design changes differ based on the dimensionality of the model the designer is using. For example, the script used for the case study involving the engine platform is different from the scripts used for the case study involving the cleaning shoe simply because one model is three-dimensional and the other is two-dimensional. The engine compartment requires a volume to be remeshed so the commands to create the grid differ from the cleaning shoe model, where vertices could be simply extruded to a specified length to create a mesh one cell thick. Once the separate scripts for two- and three-dimensional models are created, they can be modified to fit any model with a few simple changes. These changes are mostly associated with the coordinates of the area of interest within a specific model.

Within a single model, separate scripts had to be created to account for the different methods in which baffles could be created. One script dealt with baffles that were connected and created a loop using the *Close Baffles* option from the GUI, while the other handled baffles that were created either individually or were connected but not closed. The script dealing with the closed series of baffles creates a void of its own within the CFD model simply because of the way it was designed. For baffles designed individually or in a series, a thickness needs to be applied to the baffle to represent the actual thickness of a piece of sheet metal. Baffle cells within a CFD model have no thickness and although they would give a

fairly accurate solution, it is easy to add the capability to create solid cells with a small thickness from the baffle cells and is more representative of the actual model.

After the scripts mentioned above have finished running, the model is saved and the analysis is initiated. To be able to view the results from the analysis within the virtual environment, a translator must be used to transform the data written out by Star-CD™ into data that VE-Suite can recognize. The first step is to reopen Star-CD™ and get the information that the designer would like to visualize in the virtual environment such as velocity, temperature, pressure, etc. Once again, scripts containing the appropriate commands are used to gather the information from the Star-CD™ model. Several files are created from the script and used to translate the data. VE-Builder Tools is another piece of VE-Suite with many executables that allow the user to modify post-processed data for VE-Suite. One of the tools is a data translator that translates Star-CD™ data into data recognized by VE-Suite. The translator uses a parameter file, which takes the files outputted by Star-CD™ along with other input parameters to create a file using Visualization Toolkit (VTK) libraries. The translator creates a VTK file that is recognizable by VE-Suite. Once the translation is complete, a message is displayed within VE-Conductor letting the designer know that the design has been analyzed and translated and is ready to be examined.

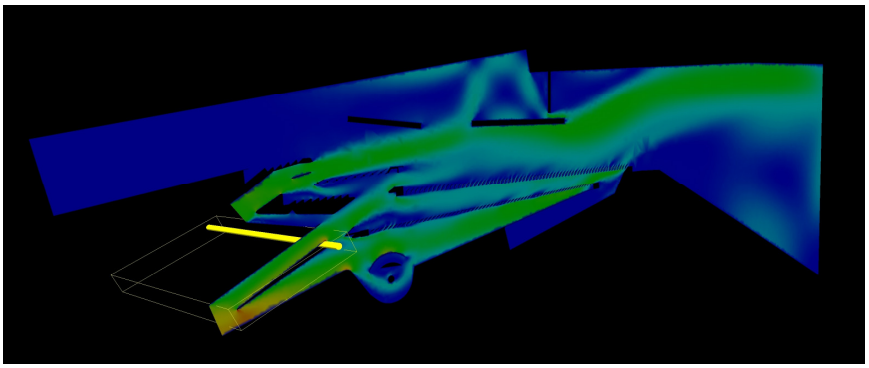

Figure 12. Velocity magnitude contour plane for cleaning shoe in VE-Suite

CHAPTER 5: RESULTS

The work completed for this thesis addresses the integration of design and analysis components into a single tool within a virtual environment. The purpose of this tool is to improve the current design and decision-making process for air flow systems which currently is inefficient in its use of time, money, and other resources. Introduction of CFD into the design process provides a powerful analysis tool, but computationally expensive simulations are common and can mean that seeing the results from one design alteration may take several days. Building multiple prototypes for field testing can also be time-consuming and expensive. This interactive baffle placement tool provides a design environment while allowing access to quick results from analyzing CFD computations that run behind the scenes.

One of the requirements of an interactive baffle placement tool is a CFD model with the capability to run simulations in a time that is acceptable to the designers. Ideally, results from analysis for each design alternative should be in real time, but with the complexity of the models examined in this research, obtaining results within an hour or two is a good starting goal given the current computation power of a designer's workstation. The harvester engine platform model originally took three days to complete an analysis of a baffle design alteration. Previous work reduced the model to the point where a simulation took three hours. Further reduction of the model and applying methods supplied within Star-CD™ allowed design alterations to be completed in 45 minutes. The harvester cleaning shoe model originally took nearly one week to complete a simulation for a baffle design change. Based on the questions that needed to be answered at this point of the design process, analysts were able to provide a two-dimensional model with a computational runtime of roughly two hours.

The final result of the interactive baffle placement tool is that it integrates design and analysis into a single environment to simplify the design and decision-making process. In the current design and decision-making process for air flow systems, the design and analysis tools are disparate. This makes it difficult for engineers with specialized backgrounds to utilize both design and analysis software. This brings about dependence between the designers and the analysts. While this relationship is beneficial and even encouraged, the level of dependence in the current design and decision-making process is excessive. In the current process, analysts rely on the designers to provide a purposed model that they prepare for analysis. The analysts prepare a full-scale model and complete the analysis for the model. For air flow systems, these models are often large and complex and take several days to complete the analysis. This means that the designers must wait several days to see the impact their design had on the air flow system. This process can be very time consuming if an acceptable design is not initially found or if designers are interested in experimenting with several designs to get a direction for the system.

Introduction of the interactive baffle placement tool into the current design and decision-making process improves the process by making it more linear. Collaboration between the designer and analyst is improved and simplified through utilizing the tool which eliminates the typical back-and-forth cycle the two. The new design and decision-making process changes the roles of both the analyst and designer. The process begins with the designers and analysts coming to a consensus about the area of interest within the system. Following the methods established in this work, the analyst can then create a single robust CFD model with interactive capabilities based on the needs of the designers. The CFD

model is then given to the designer which allows them to run simulations continually without incident. This is beneficial in many respects for both the analysts and designers. Analysts no longer have to be consulted for every design consideration, but will only need to create a single base interactive CFD model. This allows them to focus their resources on other issues within the system or in another system. Designers can now design multiple baffle configurations and view analytical results from a CFD model that runs completely behind the scenes. This means designers no longer have to wait several days for results from design changes and can view understandable analytical results without consulting an analyst. The quick turn-around time on design changes brings more analysis into the design process since more designs can be simulated and analyzed.

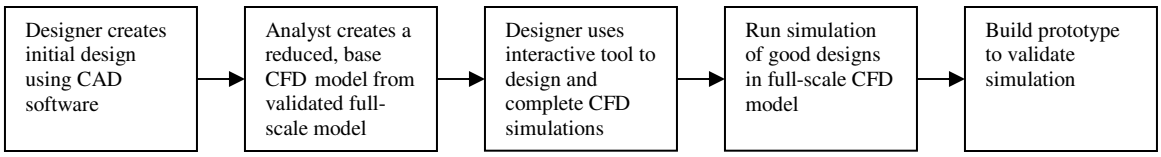

Figure 13. New design process

As the work scope evolved from the engine platform to the cleaning shoe, functionality was added to make the tool more generic for easy transition between models. Currently, the tool can be modified for a system simply by changing several values regarding the bounding box of the area of interest. These variables run throughout the code and will set the proper parameters so the bounding box can work correctly in any system. This simplifies the process for creating new models for people without coding experience who can easily enter a few global variables.

Establishing the proper techniques for creating an interactive CFD model and creating a generic interactive baffle placement tool have the final goal of getting these capabilities to

occur completely within the company. The cleaning shoe interactive baffle placement tool is already being utilized by designers, but the ability to have the designers and analysts collaborate to create new interactive baffle design tools on their own is the ideal. This decreases the company's dependence on developers to create and utilize the functionality of the interactive baffle placement tool with any system that requires baffles to redirect air.

CHAPTER 6: SUMMARY AND FUTURE WORK

The interactive baffle placement tool meets the initial goals and needs for a designer to complete design and analysis on simple baffle configurations. The integration of design and analysis capabilities into a single environment has many benefits for both the designer and analyst. Designers have the benefit of running analysis simulations and reviewing the results of their design changes without having to consult analysts. The analyst's role in the design process changes from accommodating the designer with results from every design change to creating an interactive base CFD model that allows the designer to run simulations behind the scenes without incident. This allows the analyst more time to focus on other issues or other complex models.

The ability of the interactive baffle placement tool to return analysis results in a timely manner has several benefits. Designers can design and analyze several baffle configurations daily as opposed to a few models a week. This allows the designer to create a bank of results that can be accessed anytime through VE-Suite. From this repository, designers can select several designs for further examination either on a full-scale CFD model or through the building of prototypes. In either instance, time and effort are reduced in the design process.

While all the necessary functionality to create simple baffle configurations using the interactive baffle design tool is in place, there remains interest in further improvement and extension of the tool. Currently, designers can only work with one area of interest within a given CFD model and would need several models in order to analyze each of the areas of interest within a system. Creating the functionality to allow designers to design in several locations within a CFD model would be extremely beneficial. This is simply a matter of

extending the current functionality of the GUI and creating the Star-CD™ scripts to manage all the data from each area of interest.

Along with expanding the functionality of the GUI, methods for making the GUI more generic should be also explored. It will likely take several iterations of creating new models before the interactive baffle placement tool and interactive CFD model techniques become generic enough to be modified without the developers' assistance. Eventually, the process of modifying the interactive baffle placement tool and creating interactive CFD models should be done completely in-house using the company's CFD analyst and designers.

Further research into decreasing the time required for computational analysis would allow models to be more interactive. Due to the models' complexity, assuming that these models can truly be interactive is probably illogical. However, obtaining simulation runtimes of 10 to 15 minutes is an acceptable goal. The analysis times will also improve as the computational technology continues to get better.

The current CFD models simulate the movement of single-phase air within the harvester. In reality, this is typically not the case, especially in the cleaning shoe. Many particles of varying size and density move through the cleaning shoe system. Applying multi-phase simulation capabilities has many benefits because of the actuality of the problem. Designers can see how plugging or excessive air affect the separation of the grain from other particulate.

BIBLIOGRAPHY

Bryden, K. M. and McCorkle, D. S. (2004). VE-Suite: A Foundation for Building Virtual Engineering Models of High Performance, Low Emission Power Plants. *29th International Technical Conference on Coal Utilization & Fuel Systems*. 38-46.

Cao, W., Gaertner, H., Conrad, S., Krujiff, E., Langenberg, D., and Schultz, R. (2004) Digital product development in a distributed virtual environment. *Proceedings of SPIE – The International Society for Optical Engineering, Fourth International Conference on Virtual Reality and Its Applications in Industry*. 5444 322-326.

CD Adapco Group. Star-CD Version 3.20 User Guide. (2004)

Coles, R. and Norman, E. (2005). An exploration of the role values plays in design decision-making. *International Journal of Technology and Design Education.* 15 155-171.

Comparato, J. (2003). Tutorial: Utilizing CFD results in a visualization environment. *The 2003 Electric Power Conference*, March 3-6, 2003.

Generic Math Template Library. 7 Apr. 2003. Spring 2006 http://ggt.sourceforge.net/html/main.html.

Gent, Stephen P. Incorporating computational fluid dynamics into a virtual engineering environment. Diss. Iowa State Univ., 2006.

Haque, B., Belecheanu, R., Barson, R., and Pawar, K. (2000). Towards the application of case based reasoning to decision-making in concurrent product development (concurrent engineering). *Knowledge-Based Systems.* 13, no. 2 101-112.

Huang, G., Bryden, K., and McCorkle, D. (2004). Interactive design using CFD and virtual engineering. *AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. 2 736-746.

Kenney, K., Wright, C., and Bryden, K.M. (2005). Virtual engineering approaches to developing selective harvesting technologies. Abstract submitted to ASABE 2005.

Kenney, K., Wright, C., Hoskinson, R., Hess, J.R., and Muth, D. (2006). Engineering high-fidelity residue separations for selective harvest. Abstract submitted to ASABE 2006.

Kesavadas, T. and Ernzer, M. (1999). Design of virtual factory using cell formation methodologies. *Proceedings of the AME Symposium on Virtual Reality Environments for Manufacturing*. MH 5 201-208.

Kim, S. and Ahn, B. (1999). Interactive group decision making procedure under incomplete information. *European Journal of Operational Research.* 116 498-507.

Krishnan, V. and Ulrich, K. (2001). Product development decisions: a review of the literature. *Management Science.* 47, no. 1 1-21.

McCorkle, D. S., Bryden, K. M., and Carmichael, C. G. (2003). A new methodology for evolutionary optimization of energy systems. *Comput Methods Appl Mech Eng.* 192 5021-5036.

Muth, David J. Case studies in developing comprehensive decision making environments for engineering design and analysis. Diss. Iowa State Univ., 2006.

VE-Suite. Virtual Engineering Research Group. Sept. 2007 <www.vesuite.org>.

Xiao, A., Bryden, K.M., Engelbrech, J., Huang, G., and McCorkle, D.S. (2005). Acceleration methods in the interactive design of hydraulic mixing nozzle using virtual reality tools. *Proceedings of the ASME International Mechanical Engineering Conference*.

APPENDIX: SCRIPTS FOR RUNNING STAR-CD™

Appendix A1. Star-CD™ script for running simulation without *Closed Baffle* checked on a Windows OS

```
shoe_my11_2D_c54
y
y

*set ActivateDbase 4
dbase,open,shoe_my11_2D_c54.dbs
dbase,get,4
dbase,list,4,,Long

cset,all
vset,all
cdel,cset
vdel,vset
vread,InteractiveBaffle_0.vrt,0,,,code
cread,InteractiveBaffle_0.cel,0,,,add,code,,
vread,InteractiveBaffle_1.vrt,143,,,code
cread,InteractiveBaffle_1.cel,143,,,add,code,,
vread,InteractiveBaffle_2.vrt,286,,,code
cread,InteractiveBaffle_2.cel,286,,,add,code,,
vread,InteractiveBaffle_3.vrt,429,,,code
cread,InteractiveBaffle_3.cel,429,,,add,code,,
cset,all
ctyp,10
cmod,cset
corient,cset,,

vset,all
vmerge,vset,,0.01
vset,all
vcex,1,,cset,,,normal,1,0,both,uniform

cset,news,shell
cdel,cset
cset,all
live,surface,create
cset,news,fluid
cdel,cset
cset,all
```

Appendix A1. (continued)

```
cset,news,grange,1300,2100,-190,-170,-800,-300
cset,add,grange,1300,2100,-210,-190,-800,-300

cdel,cset
cset,all
vset,all
ccomp,cset
y
vcomp,vset
y

dbase,open,shoe_my11_2D_c54.dbs
dbase,add,3,1000,,,
vset,all
vmerge,vset,,0.01
cset,all
vset,news,grange,1300,2100,-190,-170,-800,-300
vset,news,vlist,10,1031
hfil,vset,cset

cset,news,grange,1300,2100,-190,-170,-800,-300
local,10,cart,,,,,,,
vloc,10,cart,1001,Y,1003,Z,1139
csys,10
vcex,1,,cset,,,local,25,0,0,both,uniform

cset,news,shell
cdel,cset
cset,all

dbase,open,shoe_my11_2D_c54.dbs
dbase,put,5,star,over
dbase,list,5,,Long
tcl,done

dbase,open,shoe_my11_2D_c54.dbs
dbase,add,2,,10000,0,
cset,,

dbase,open,shoe_my11_2D_c54.dbs
dbase,put,23,star,over
Complete Assembly
dbase,list,23,,Long
```

Appendix A1. (continued)

```
tcl,done

check,cset,,right,newset,nolist
cflip,cset

cset,all
vset,all
vmerg,vset
cset,all
vset,all

memo,maxcut,3000000

cset,news,grange,1300,2100,-220,-170,-850,-300

cpta,10,1,2,4,3,off
cptn,10,complete_mesh
cpty,10
cpcr,cset,10
cset,all

bset,news,region,6
bdel,bset
bset,news,region,7
bdel,bset

vset,news,edge
memo,maxnbu,400000
bfin,6,1001
bfin,7,1002
rdef,6,symplane,standard
rdef,7,symplane,standard

cset,all
vset,all
bset,all
ccomp,cset
y
vcomp,vset
y

pmat,1,fluid
cset,news,grange,3288.3,3292.5,-202,-170,-264.9,-260.0
```

Appendix A1. (continued)

```
*get,monicellnum,cset,1
moni,monicellnum

cset,news,grange,7477.05,7500,-202,-170,45.8,86
*get,prescellnum,cset,1
pres,1.e+05,prescellnum


cset,all

pmat,1,fluid,AIR,0
turb,ke,1,stan
lowre,off
coke,0.09,1.44,1.92,1.44,-0.33,0.419,1,1.219,0.9
pmat,1,fluid
cini,y,y,,0
init,stan,0,0,0,1,0,0,mixl,0.001,0.001,293
solve,LU,y
solve,LV,n
solve,LW,y
solve,LP,y
solve,LKE,y
solve,LEPS,y
solve,LT,n
solve,LVIS,y
solve,LDEN,n
solve,LLAMV,n
solve,LCP,n
solve,LCOND,n
solve,LUU,n
solve,LVV,n
solve,LWW,n
solve,LUV,n
solve,LVW,n
solve,LUW,n
solve,LV22,n
solve,LF22,n
solve,LA2,n

relax,0.5,0.2,0.5,,1,,,,,,,,
sweep,100,100,100,1000,100,100,,,,,,,,,,,
resid,0.1,,0.1,0.05,0.1,0.1,,,,,,,,,,,
dsch,mars,uvw,0.5,stan
```

```
dsch,mars,turb,0.5,stan

iter,2000,0.001
save,shoe_my11_2D_c54.mdl
geom,shoe_my11_2D_c54.geom,0.001,bina,chec
save,shoe_my11_2D_c54.mdl
prob,shoe_my11_2D_c54.prob,binary
quit,save
```

Appendix A2. Star-CD™ script for running simulation with *Closed Baffle* checked on a Windows OS

```
shoe_my11_2D_c54
y
n

*set ActivateDbase 4
dbase,open,shoe_my11_2D_c54.dbs
dbase,get,4
dbase,list,4,,Long

cset,all
vset,all
cdel,cset
vdel,vset
vread,InteractiveBaffle_0.vrt,0,,,code
cread,InteractiveBaffle_0.cel,0,,,add,code,,
vread,InteractiveBaffle_1.vrt,143,,,code
cread,InteractiveBaffle_1.cel,143,,,add,code,,
vread,InteractiveBaffle_2.vrt,286,,,code
cread,InteractiveBaffle_2.cel,286,,,add,code,,
vread,InteractiveBaffle_3.vrt,429,,,code
cread,InteractiveBaffle_3.cel,429,,,add,code,,
cset,all
ctab,10,shell,0,,,,,,,off,
ctyp,10
cmod,cset

ccomp,cset
y
vcomp,vset
y

dbase,open,shoe_my11_2D_c54.dbs
dbase,add,3,1000,,,

vset,all
vmerge,vset,,0.01
cset,all
vset,news,grange,1300,2100,-190,-170,-800,-300
vset,news,vlist,10,1031
hfil,vset,cset
```

Appendix A2. (continued)

```
cset,news,grange,1300,2100,-190,-170,-800,-300
local,10,cart,,,,,,,
vloc,10,cart,1001,Y,1003,Z,1139

csys,10
vcex,1,,cset,,,local,25,0,0,both,uniform

cset,news,shell
cdel,cset
cset,all

dbase,open,shoe_my11_2D_c54.dbs
dbase,put,5,star,over
dbase,list,5,,Long
tcl,done

dbase,open,shoe_my11_2D_c54.dbs
dbase,add,2,,5000,0,
cset,,

dbase,open,shoe_my11_2D_c54.dbs
dbase,put,23,star,over
Complete Assembly
dbase,list,23,,Long
tcl,done

check,cset,,right,newset,nolist
cflip,cset
cset,all

vset,all
vmerg,vset
cset,all
vset,all

memo,maxcut,3000000
cset,news,grange,1300,2100,-220,-170,-850,-300

cpta,10,1,2,4,3,off
cptn,10,complete_mesh
cpty,10
cpcr,cset,10
```

Appendix A2. (continued)

```
cset,all

bset,news,region,6
bdel,bset

bset,news,region,7
bdel,bset

vset,news,edge
memo,maxnbu,400000
bfin,6,1001
bfin,7,1002
rdef,6,symplane,standard
rdef,7,symplane,standard

cset,all
vset,all
bset,all

ccomp,cset
y
vcomp,vset
y

pmat,1,fluid
cset,news,grange,3288.3,3292.5,-202,-170,-264.9,-260.0
*get,monicellnum,cset,1
moni,monicellnum

cset,news,grange,7477.05,7500,-202,-170,45.8,86
*get,prescellnum,cset,1
pres,1.e+05,prescellnum

cset,all

pmat,1,fluid,AIR,0
turb,ke,1,stan
lowre,off
coke,0.09,1.44,1.92,1.44,-0.33,0.419,1,1.219,0.9
pmat,1,fluid
cini,y,y,,0
init,stan,0,0,0,1,0,0,mixl,0.001,0.001,293
solve,LU,y
```

Appendix A2. (continued)

```
solve,LV,n
solve,LW,y
solve,LP,y
solve,LKE,y
solve,LEPS,y
solve,LT,n


solve,LVIS,y
solve,LDEN,n
solve,LLAMV,n
solve,LCP,n
solve,LCOND,n
solve,LUU,n
solve,LVV,n
solve,LWW,n
solve,LUV,n
solve,LVW,n
solve,LUW,n
solve,LV22,n
solve,LF22,n
solve,LA2,n

relax,0.5,0.2,0.5,,1,,,,,,,,
sweep,100,100,100,1000,100,100,,,,,,,,,,,,
resid,0.1,,0.1,0.05,0.1,0.1,,,,,,,,,,,,

dsch,mars,uvw,0.5,stan
dsch,mars,turb,0.5,stan

iter,2000,0.001

save,shoe_my11_2D_c54.mdl
geom,shoe_my11_2D_c54.geom,0.001,bina,chec
save,shoe_my11_2D_c54.mdl
prob,shoe_my11_2D_c54.prob,binary

quit,save
```

Appendix A3. Star-CD™ script for gather post-processing information

```
shoe_my11_2D_c54
y
y

load,shoe_my11_2D_c54.pst
memo maxsc2 200000000
memo maxcel 10000000

cset,all
ccomp
y
vset,all
vcomp,vset
y

cset,news,fluid
vset,news,cset
cwrite,shoe_my11_2D_c54.cel,,cset,,coded,
vwrite,shoe_my11_2D_c54.vrt,,vset,coded

oper,getv,su,1
oper,getv,sv,2
oper,getv,sw,3
oper,getv,p,4,relative,1
oper,getv,t,5,relative,1
oper,getv,te,6
savu,shoe_my11_2D_c54.usr,all,coded,all

close shoe_my11_2D_c54.usr
cdsa,shoe_my11_2D_c54.cel,shoe_my11_2D_c54.vrt,-1,shoe_my11_2D_c54.inp,-1
close all

quit,nosave
```

Appendix A4. Code to create cell and vertex files representing baffles in Star-CD™

```
#include "tets.h"
#include <iostream>
#include <iomanip>

using namespace std;
using namespace gmtl;
#define PI 3.14159

Tets::Tets()
{
}
Tets::~Tets()
{
}

using namespace std;

void Tets::Baffle( std::vector< double > baffleOneParams,
                   std::vector< double > baffleTwoParams,
                   std::vector< double > baffleThreeParams,
                   std::vector< double > baffleFourParams )
{
   std::ostringstream baffleFileStream;
   baffleFileStream << "BaffleParameters.txt";
   std::string baffleParamString = baffleFileStream.str();

   ofstream outputBaffleParams;
   outputBaffleParams.open( baffleParamString.c_str() );

   outputBaffleParams<<"Baffle  Length  x-location  y-location  z-location
                        x-angle  y-angle  z-angle"<<"\n";
   for( size_t i = 0; i < 4; ++i )
   {
      if( i == 0 )
      {
         baffleParams = baffleOneParams;
         m_baffleNumber = "1";
      }
      else if( i == 1 )
      {
         baffleParams = baffleTwoParams;
         m_baffleNumber = "2";
      }
```

Appendix A4. (continued)

```
else if( i == 2 )
      {
         baffleParams = baffleThreeParams;
         m_baffleNumber = "3";
      }
      else if( i == 3 )
      {
         baffleParams = baffleFourParams;
         m_baffleNumber = "4";
      }

      if( baffleParams.empty() )
      {
         return;
      }
      double plane = baffleParams.at( 0 );
      double height = baffleParams.at( 1 );
      double width = baffleParams.at( 2 );
      double xLocation = baffleParams.at( 3 );
      double yLocation = 25;
      double zLocation = baffleParams.at( 5 );
      double xTheta = baffleParams.at( 6 );
      double yTheta = baffleParams.at( 7 );
      double zTheta = baffleParams.at( 8 );

      outputBaffleParams.setf(ios::showpoint);
      outputBaffleParams<<m_baffleNumber;
      outputBaffleParams<<setprecision(7)<<setw(17)<<height;
      outputBaffleParams<<setprecision(7)<<setw(11)<<xLocation;
      outputBaffleParams<<setprecision(7)<<setw(14)<<yLocation;
      outputBaffleParams<<setprecision(7)<<setw(14)<<zLocation;
      outputBaffleParams<<setw(15)<<xTheta;
      outputBaffleParams<<setw(10)<<yTheta;
      outputBaffleParams<<setw(11)<<zTheta<<"\n";


      int vertNum = 0;

      long int cellNum = vertNum;
      int cellType = 5;
      int cellBaffle = 3;
```

Appendix A4. (continued)

```
numYBaf = 1;
numZBaf = height / 5;

if( numZBaf == 0 )
{
   numZBaf = 1;
}

numYVerts = 1 + numYBaf;
numZVerts = 1 + numZBaf;

totalVert = numYVerts * numZVerts;
totalBaf = numYBaf * numZBaf;

double** Point = new double*[totalVert];
for(int i=0; i<totalVert; i++)
{
   Point[i] = new double[3];
}

double*** vertices = new double**[numYVerts];
for(int i=0; i<numYVerts; i++)
{
        vertices[i] = new double*[numZVerts];
            for(int j=0; j<numZVerts; j++)
            {
                    vertices[i][j] = new double[4];
            }
}

Matrix44f worldMat;//create an identity matrix

gmtl::Point3f trans;
gmtl::Point3f startLocation(static_cast <float>(xLocation),
                            static_cast <float>(-176.1),
                            static_cast <float>(zLocation));
// translate world dcs by distance that the head
// is away from the origin
gmtl::Matrix44f transMat = gmtl::makeTrans< gmtl::Matrix44f >( startLocation );
gmtl::Matrix44f baffleTrans = transMat * worldMat;
gmtl::Point3f newJugglerHeadPoint;
// get the position of the starting location
gmtl::Point3f Points = baffleTrans * newJugglerHeadPoint;
```

Appendix A4. (continued)

```
//create vertices in baffle
if( plane == 1 )
{
   for( int i=0; i<numYVerts; i++ )

  {
      for( int j=0; j<numZVerts; j++ )
      {
         Point[i*numZVerts+j][0] = 0;
         Point[i*numZVerts+j][1] = -25 * i;
         Point[i*numZVerts+j][2] = height / numZBaf * j;
      }
   }
}
//make a rotation matrix
gmtl::EulerAngleXYZf baffleRotation(static_cast <float>(xTheta*PI/180),
                                    static_cast <float>(yTheta*PI/180),
                                    static_cast <float>(zTheta*PI/180));

gmtl::Matrix44f rotMatrix = gmtl::makeRot< gmtl::Matrix44f >( baffleRotation );
gmtl::Vec4f vertexPos[282];

for( int i=0; i<totalVert; i++ )
{
   vertexPos[i][ 0 ] = static_cast <float>(Point[i][ 0 ]);
   vertexPos[i][ 1 ] = static_cast <float>(Point[i][ 1 ]);
   vertexPos[i][ 2 ] = static_cast <float>(Point[i][ 2 ]);
}

gmtl::Vec4f rotatePoints[282];
gmtl::Vec4f finalPoints[282];

//rotate and translate vertices
for( int i=0; i<totalVert; i++ )
{
   rotatePoints[i] = rotMatrix * vertexPos[i];
   finalPoints[i][0] = Points[0] + rotatePoints[i][0];
   finalPoints[i][1] = Points[1] + rotatePoints[i][1];
   finalPoints[i][2] = Points[2] + rotatePoints[i][2];
}

std::ostringstream vrtFileStream;
vrtFileStream << "InteractiveBaffle_" << i << ".vrt";
```

Appendix A4. (continued)

```
std::string vrtFileString = vrtFileStream.str();

ofstream outputVertexFile;
outputVertexFile.open( vrtFileString.c_str() );

std::ostringstream celFileStream;

celFileStream << "InteractiveBaffle_" << i << ".cel";
std::string celFileString = celFileStream.str();

ofstream outputCellFile;
outputCellFile.open( celFileString.c_str() );
for(int i=0; i<numYVerts; i++)
{
   for(int j=0; j<numZVerts; j++)
   {
            vertices[i][j][0] = vertNum + 1;
                  vertices[i][j][1] = finalPoints[i*numZVerts+j][0];
                  vertices[i][j][2] = finalPoints[i*numZVerts+j][1];
                  vertices[i][j][3] = finalPoints[i*numZVerts+j][2];

      if( vertices[i][j][2] == 0 )
      {
         outputVertexFile.unsetf( ios::showpoint );
                        outputVertexFile << setw(9) << vertices[i][j][0];
         outputVertexFile.setf( ios::showpoint );
         outputVertexFile << "       " << setprecision( 9 ) << setw(10)
                                                << vertices[i][j][1]\
                          << "     " << setprecision( 10 ) << setw(10)
                                                << vertices[i][j][2]\
                          << "      " << setprecision( 9 ) << setw(10)
                                                << vertices[i][j][3] << "\n";
      }
      else if( vertices[i][j][2] <= 0 )
      {
         outputVertexFile.unsetf( ios::showpoint );
         outputVertexFile << setw(9) << vertices[i][j][0];
         outputVertexFile.setf( ios::showpoint );
         outputVertexFile << "       " << setprecision( 9 ) << setw(10)
                                                << vertices[i][j][1]\
                          << "     " << setw(10) << vertices[i][j][2]\
                          << "      " << setw(10) << vertices[i][j][3] << "\n";
      }
```

Appendix A4. (continued)

```
            else
            {
               outputVertexFile.unsetf( ios::showpoint );
               outputVertexFile << setw(9) << vertices[i][j][0];
               outputVertexFile.setf( ios::showpoint );
               outputVertexFile << "        " << setprecision( 9 ) << setw(10)
                                                                   << vertices[i][j][1]\

                                << "      " << setw(10) << vertices[i][j][2]\
                                << "      " << setw(10) << vertices[i][j][3] << "\n";
            }
            vertNum = vertNum + 1;
         }
      }

      outputVertexFile.close();
      double vertex1;
      double vertex2;
      double vertex3;
      double vertex4;
      for( int i=0; i<numYBaf; i++)
      {
         for(int j=0; j<numZBaf; j++ )
         {
            cellNum = cellNum + 1;
            vertex1 = vertices[i][j][0];
            vertex2 = vertices[i][j][0] + 1;
            vertex3 = vertices[i][j][0] + numZVerts + 1;
            vertex4 = vertices[i][j][0] + numZVerts;

            outputCellFile << setw( 9 ) << cellNum << setw( 9 ) << vertex1 << setw( 9 )
                                                                          << vertex2\
                           << setw( 9 ) << vertex3 << setw( 9 ) << vertex4 << setw( 9 )
                                                                          << 0\
                           << setw( 9 ) << 0 << setw( 9 ) << 0 << setw( 9 ) << 0
                                                                          << setw( 9 )\
                           << cellType << setw( 9 ) << cellBaffle << "\n";
         }
      }
      outputCellFile.close();
   }
   outputBaffleParams.close();
}
```