Abstract: The difference between a computer game and a simulator can be a small one—both require the same capabilities from the computer: realistic graphics, behavior consistent with the laws of physics, a variety of scenarios where difficulties can emerge, and some assessment technique to inform users of performance. Computer games are a multi-billion dollar industry in the United States, and as the production costs and complexity of games have increased, so has the effort to make their creation easier. Commercial software products have been developed to greatly simplify the game-making process, allowing developers to focus on content rather than on programming. This paper investigates Unity3D game creation software for making three-dimensional engine-room simulators. Unity3D is arguably the best software product for game creation, and has been used for numerous popular and successful commercial games. Maritime universities could greatly benefit from making custom simulators to fit specific applications and requirements, as well as from reducing the cost of purchasing simulators. We use Unity3D to make a three-dimensional steam turbine simulator that achieves a high degree of realism. The user can walk around the turbine, open and close valves, activate pumps, and run the turbine. Turbine operating parameters such as RPM, condenser vacuum, lube oil temperature, and governor status are monitored. In addition, the program keeps a log of any errors made by the operator. We find that with the use of Unity3D, students and faculty are able to make custom three-dimensional ship and engine room simulators that can be used as training and evaluation tools.

Keywords: simulators, PC simulators, game-programming
A 3-dimensional engine simulator is not very different from a game: like the game, the simulation must behave in a physically realistic way, respond to user inputs, allow motion in the 3-D space, and advise the user of the status of the simulation (pressures, temperatures, etc.).

Creating a PC-based simulator or a game from the ground-up is not an easy task.

Computer games and simulators are typically written in C++ or some other programming language. The seemingly simple act of displaying a 2-dimensional image on a PC screen requires about 200 lines of code in C++ [1]. The problem is that a game or simulator must perform two non-trivial tasks: deal with Windows, and display and move objects in three dimensions.

To aid programmers in developing PC games, powerful software products have been developed which can greatly simplify the programming task. These programs have pre-made functions or subroutines that handle some of the difficult parts of creating 3D simulators or games, and allow the programmer to focus on the content of the simulation.

Some of these products are free, such as Lite-C [2], Panda3D [3] and Blender [4], while most are commercial products, such as DarkBasic [5], Torque [6] and Blitz 3D [7]. Readers are referred to Wikipedia [8] for a list of software products which can be used to make games and simulators.

Perez [9] showed that game-making software product Jamagic could be used for making a variety of 3-D simulators, including ship handling and flight simulations. Jamagic was also used to make a 2-D engine room simulator, presented at the 9th International Conference of Engine Room Simulators (ICERS 9) by Perez and Byra [10]. The authors showed that game-making software could be used to make a 2-Dimensional panel-type simulator of a steam turbine.

Some of the products mentioned above use their own or simple programming languages, for example DarkBasic, Blender and Blitz3D, while most require writing code in C++ or Python language to access pre-programmed routines for making games (known as game engines).

Using game programming software, the task of making a simulator becomes greatly simplified, enabling learning institutions to make their own simulators. An important aspect of this, aside from the obvious one of cost, is the ability to tailor-fit assessment techniques to the needs of individual institutions. However, it must be remembered that creating a simulator is not an easy task, as algorithms still must be created to model the physics involved with engine room simulators.

In this paper we explore the use of Unity software (www.unity3d.com) for 3-D modeling of a small steam turbine, including user assessment.

Unity is an extremely powerful tool for creating 3D video games, and can run on Microsoft Windows and Mac OS X. Many highly successful commercial games have been created with Unity. The games it produces can be run on Windows, Mac, Xbox 360, PlayStation 3, Wii, iPad, iPhone, as well as the Android platform.

The software version used for this simulation is freely available from the Unity web site – this is a fully functional version of the Pro Version which costs about $1500.

2. 3-D Art

One of the first steps in making a game or simulator is obtaining the 3-dimensional images of the simulated object. For this project we needed a realistic 3-dimensional model of a steam turbine, with valves, pipes, and displays.

Usually one has to make the 3D objects using commercially available software like Maya, Studio Max, or the freeware Blender. Making 3D game objects can be a very time-consuming
task, but occasionally the 3-D objects can be found on the internet already made – these objects may be ready for use, or may be modified using 3-D modeling software.

We were fortunate to find 3-D art of turbines, pipes and valves that was sufficiently similar to actual systems to greatly simplify our task. The turbine model came with no piping, but the pipe models used were easy to modify and add to the 3-D scene.

Once the 3-D art is available, it can easily be “dragged” into the Unity scene.

3. Programming

Any simulator must be given instructions how to behave. For example: what happens when a pump is activated or a valve opened? How much does the pressure drop across a valve while throttling? How does one control the pitch of the turbine sound as the RPM change? By writing a computer program, developers can control the action on the simulation.

Unity can be programmed in either Java or C# - the programmer selects which. For this paper Java was selected because it is somewhat simpler to use.

A sample portion of the main program used for this project is shown below. It illustrates an algorithm used to control the pitch and speed of the turbine based on the amount that two valves upstream of the turbine are opened. The simulator created for this project required about 500 lines of program.

```java
rate = spin1/90*spin3/90; // rate depends on valve 1 and valve 3 open amount
rpmgoal = 1250*rate;
if (rpm>0)
{
    audio.pitch = 0.1 + rpm/300.0; // adjust the pitch based on rpm
    if (tsound<0)
    {
        audio.clip=turbsound; // the turbine sound clip to use
        audio.Play(); // play the clip
        tsound = 1.0;
    }
}
if (rpmgoal>rpm & rpm<900) // we add to rpm
{
    rpm = rpm + Time.deltaTime*(-spin3)/7;
}
```

4. The Simulation

The simulation developed for this project is based on a steam-driven turbine/generator at the U.S. Merchant Marine Academy. The goal was to demonstrate the capabilities of Unity for engine room simulations and evaluation of student performance.

The model displays operating parameters such as turbine RPM, condenser vacuum, lube oil temperature, and governor status. A list of errors is displayed on the screen informing students and instructors of any missteps.
The operator must perform tasks in the proper in order to avoid error messages. Before admitting steam into the turbine the following tasks must be performed: oil must be checked, gland-sealing steam must be admitted into turbine, the lube oil pump must be primed, vacuum raised in the condenser, and sea water and condensate pumps turned on.

In addition, a drain must be opened in the main steam line before opening the throttle; this allows steam to be visible exiting the drain so the operator can visually check the flow of steam for presence of liquid droplets.

Once the turbine begins rolling after the throttle is opened, the pitch of the turbine sound will rise as RPM increase. As the throttle is opened or closed, turbine RPM reacts accordingly. Also, the lube oil primer will shut itself off at a given value of the RPM.

The status of the turbine is displayed in a 3D text above the turbine, while the status of all pumps is shown in a pump menu at the bottom of the screen.

In order to aid students in understanding the system, the user can right-click on objects to display a message describing the item clicked. Valves are opened by left-clicking (more clicks open the valves more) and closed by right-clicking. The camera (which provides the view to the user) is moved using the arrow keys (Table 1) as well as by number keys:

<table>
<thead>
<tr>
<th>Key</th>
<th>Camera action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Rotate left</td>
</tr>
<tr>
<td>2</td>
<td>Rotate right</td>
</tr>
<tr>
<td>3</td>
<td>Move down</td>
</tr>
<tr>
<td>4</td>
<td>Move right</td>
</tr>
<tr>
<td>5</td>
<td>Rotate down</td>
</tr>
<tr>
<td>6</td>
<td>Rotate up</td>
</tr>
</tbody>
</table>

The screenshots below show the simulator in operation. The user has the capability to walk around the 3D space at will using the arrow keys and the number keys listed above.
Figure 1. The turbine and generator.

Figure 1 shows the turbine with associated piping and valves. The vacuum, seawater circulation, condensate, and oil primer pumps are controlled by the buttons at the bottom of the scene. Turbine operating parameters are displayed over the turbine. The valves can be clicked to open and close.

Figure 2. Turbine Close-up
Three-Dimensional Engine Simulators with Unity3D Game Software

Figure 2 is a close-up of the turbine. The upper left valve is the throttle, the middle valve is the steam drain, and the lower valve allows gland-sealing steam into the turbine. The two pipes exiting the turbine are the gland sealing steam inlet and outlet lines.

When the throttle is opened, the turbine slowly accelerates, and reaches a RPM value based on the amount the throttle and main steam valve are open. As the turbine turns faster, the pitch of the engine sound increases. Shutting the throttle down causes the pitch and RPM to drop gradually.

Figure 3. The steam drain line. Steam can be seen exiting the pipe.

When the steam drain line is open, steam is seen exiting the pipe as a fog. As the pipe is opened more, the steam exits further from the pipe, and becomes clear close to the pipe, indicating superheated steam at the exit.
Figure 4 shows the turbine from the side, with the generator connected to it.

Figure 5 shows three of the pump controls. The controls are a red color when pumps are off, and turn blue when activated. The lube oil primer pump shuts itself off automatically at a set RPM, and the lube oil pump starts.
Figure 6. Display of error messages.

Figure 6 shows sample error messages displayed on the screen. The error messages are activated by a number of errors, including the failure of the operator to open the throttle slowly.

5. Observations and Conclusions

Unity is an extremely powerful and versatile tool that can create simulators as good as any commercial product on the market – in fact, Unity has been used to make many successful commercial games. Unity’s ability to quickly render 3-D scenes and execute code makes it ideal for simulators as well as games. The entire engine room of a ship could potentially be simulated using the software, with much less work than if the simulation were programmed directly using a programming language.

Unity has a rather steep learning curve, especially if the programmer is not accustomed to object-oriented languages. However, the on-line documentation is very good, and the user forums are of immense value in learning how to do specific tasks – one merely has to type what one wishes to learn into a search engine, and usually someone else has already answered that same question in the past.

The project was completed with 60 -100 hours of work, with the programmer already somewhat familiar with Unity. Quite a bit of time was saved by purchasing pre-made 3D models of the turbine and piping systems, which came to about $50 total. It is anticipated that the time required for any future projects of similar complexity would be considerably less due to the programmer being more familiar with the software.

References
