Final Technical Report
TNW2005-06
TransNow Budget 62-8990
Virtual Training Tools for Transportation Infrastructure Construction

Virtual Training Tools for Transportation Infrastructure Construction

by

George M. Turkiyyah
Joe P. Mahoney
Drew Batchelor

University of Washington
Department of Civil and Environmental Engineering
Seattle, WA

Report prepared for:
Transportation Northwest (TransNow)
Department of Civil Engineering
129 More Hall
University of Washington, Box 352700
Seattle, WA 98195-2700

March 2005

TECHNICAL REPORT STANDARD TITLE PAGE

1. REPORT NO. TNW2005-06	2. GOVERNMENT ACCESSION NO.	3. RECIPIENT'S CATALOG NO.

4. TITLE AND SUBTITLE VIRTUAL TRAINING TOOLS FOR TRANSPORTATION INFRASTRUCTURE CONSTRUCTION	5.REPORT DATE March 2005 6. PERFORMING ORGANIZATION CODE
7. AUTHOR(S) George Turkiyyah, Joe P. Mahoney, Drew Batchelor	8. PERFORMING ORGANIZATION REPORT NO. TNW2005-06
9. PERFORMING ORGANIZATION NAME AND ADDRESS Transportation Northwest Regional Center X (TransNow) Box 352700, 123 More Hall University of Washington Seattle, WA 98195-2700	10. WORK UNIT NO. 11. CONTRACT OR GRANT NO. DTRS99-G-0010
12. SPONSORING AGENCY NAME AND ADDRESS United States Department of Transportation Office of the Secretary of Transportation 400 Seventh St. SW Washington, DC 20590	13. TYPE OF REPORT AND PERIOD COVERED Final Report 14. SPONSORING AGENCY CODE

15. SUPPLEMENTARY NOTES

This study was conducted in cooperation with University of Washington.

16. ABSTRACT

This project's ultimate goal was to bring cost-effective interactive 3D training environments to contractors and state agencies. This was accomplished by developing and testing a second version of the XPactor training simulation software. While the original XPactor simulator focused exclusively on a single compactor operator in a single paving scenario, the enhancements expand the capabilities of the previous system and work towards creating a virtual hot mix asphalt (HMA) paving construction site. This site will allow multiple users to interact with one another and assume a variety of key construction roles including paver operator, multiple types of compactor operators, site coordinator, and even public motorist. The virtual site is configurable to present users with a rich set of construction projects, such as paving different geometries like multilane, curved, and inclined roads, and paving under different environmental conditions including weather, time of day (night), and traffic patterns. As with the current version, the trainer will be able to set these parameters at runtime in order to customize training.

17. KEY WORDS Pavement, Simulation, Software, Construction	18. DISTRIBUTION STATEMENT No restrictions. This document is available to the public through the National Technical Information Service, Springfield, VA 22616

19. SECURITY CLASSIF. (of this report) None	20. SECURITY CLASSIF. (of this page) None	21. NO. OF PAGES 29	22. PRICE $8.75

DISCLAIMER

Table of Contents

1. Introduction 5
2. Problem Statement 5
3. Objectives 6
4. Implementation Details 7
    4.1 General Implementation Notes 7
    4.2 Scenario Implementation 8
        4.2.1 The Scenario XML File Format 9
        4.2.2 The Static Environment 10
        4.2.3 The Road 13
        4.2.4 Road Segments 14
        4.2.5 The Roller 18
        4.2.6 The Paver 19
    4.3 Network Implementation 20
        4.3.1 The Simulation Server 21
        4.3.2 The Network Protocol 22
        4.3.3 Update Timing 23
        4.3.4 Synchronizing the Rollers 23
        4.3.5 Syncronizing the Paver 23
        4.3.6 Synchronizing the Temperature 24
5. Summary and Future Work 24
Appendix A. Walkthrough and User's manual 25
    Starting a New Simulation 25
    Single User Simulation 25
    Start New Network Simulation 25
    Join Existing Network Simulation 26
    Viewing the world 26
    Running the Paver 26
    Driving the Roller 27
    Text Messaging 29

1. Introduction

Pavement construction is a complex, multi-person task that requires careful coordination and proper timing between a sequence of activities ranging from bringing hotmix to the sit, laying it on the road being constructed, and compacting it through multiple passes before it reaches a critical cold temperature. Improper execution results both in inefficiencies in the construction process and, more importantly, results in pavements that deteriorate prior to their design life. Given the amount of hotmix asphalt pavements built every year, the cost of inferior construction translates to millions of dollars annually.

Tools that assist people in visualizing and understanding the constraints on the construction process are needed. Current computer technology has made it possible to develop three dimensional simulations that provide immersive and interactive training environments where users can perform and experience pavement construction tasks. These environments allow navigation in a realistic 3D world, bring in the time dimension of hotmix laydown and cooling, and allow users to perform construction activities and see their effects in real time. In order for such simulations to more realistically portray real construction scenarios, multiple users need to be able to interact simultaneously with the environment and observe each others actions and react appropriately.

These interactive training simulations offer multiple benefit to hotmix construction. They allow a novice to practice construction tasks in a virtual environment before they execute them on site. Trainees have the opportunity to operate equipment, experience the sequence events in a construction scenario, become sensitive to the timing constraints of a given task, and interact with other participants in the construction process. In addition, the combination of an accurate geometric representation of the construction site, the time-constraints derived from the hotmix cooling physics, and the multi-user interaction of a simulation, gives construction managers insights into the planning, timing, potential bottlenecks, and communication needs of a construction project.

2. Problem Statement

The goal of this project was to develop a second version of the XPactor training simulation software. While the original XPactor simulator focused exclusively on a single compactor operator in a single paving scenario, the proposed enhancements expand the capabilities of the current system and work towards creating a virtual hot mix asphalt (HMA) paving construction site.

A key element of the virtual construction site is the ability for multiple users to interact within the same simulation simultaneously. Users can play the role of a roller operator, with multiple rollers allowed in a single simulation. Users also can control the paving machine, dictating the timing of the hotmix laydown, with the goal of coordinating with the rollers to ensure that the asphalt can be compacted within the cooling time window. Finally, users can participate in the simulation as observers that can monitor the activities of the other participants and direct and advise them if needed. This multi-user interaction is achieved by making the simulation software network-enabled. Participants run the software on separate computers communicating over a network.

For the simulation software to be broadly applicable, it should provide the functionality to present various paving scenarios. Elements of the simulation that need to be flexible include the layout of the road to be paved, the number of participants in the construction operation, the environmental parameters that affect the cooling of the hotmix, and the more general environment in which the simulation takes place. These flexible elements must be configurable by the users or administrators of the simulation and should not require modifications to the simulation software. Allowing users to create and modify their own paving scenarios ensures that trainers and planners can use the simulation for customized training, applicable to their specific objectives.

3. Objectives

The specific objectives of this project include the following.

- Produce a real-time simulation for pavement construction training. The specific construction task that this simulation addresses is hotmix laydown and compaction. This includes paving machines and rollers, the timing of asphalt laydown, the temperature changes of the hotmix, and the 3D geometry of the operating environment. Performance feedback is provided to user on the completion of road compaction which reflects both the efficiency of the roller operation and the timing of the hotmix laydown by the paver.
- Make the simulation network-enabled so that multiple users can simultaneously participate in real-time, with each user assuming a different role. These interactions must be synchronized such that the state of the construction project can is current for all users. Additionally, functionality should be included to allow the users to communicate with each other over the network during the simulation in order to coordinate the construction process.

- The simulation should be able to display a wide variety of paving scenarios. This should include the ability to use different road shapes and sizes, work with varying numbers of users and construction roles, and set various environmental conditions affecting pavement cooling time. The customizable elements of a scenario should be exposed via XML to provide a framework, allowing end users to develop their own construction scenarios.

We have developed a roller simulation software application that meets these objectives. The system provides a virtual paver and one or more virtual compactors interacting in a simulated pavement construction site. Users can take on multiple roles in the simulation including controlling the laydown of hotmix, and driving the rollers in order to achieve optimal pavement compaction. Visual feedback provides the user with real time information as to the current temperature of the cooling pavement and the degree of compaction of the road. The simulation can be run either in a stand-alone single-user mode or, by using the provided simulation server, multiple users can participate in the simulation over a network. The network simulation includes a passive observer role and a text messaging feature that allows the participants to communicate over the network. The system can be downloaded from http://cae4.ce.washington.edu/vtools. A walkthrough of the system and user's manual are included in Appendix A of this report. The rest of this report describes the lower level implementation details of the software and concludes with areas of future work.

4. Implementation Details

This section describes the details of the implementation of the pavement compaction simulator.

4.1 General Implementation Notes

The release of Microsoft's DirectX 9 3D API included libraries allowing the development of .NET framework-based applications. Using the .NET framework and C # in particular, resulted in a significantly lower cost of development than creating a similar application is native code. This ease of development comes at the cost of a slight performance hit, but upon evaluating the performance benchmarks; we decided that this hit was acceptable as it allowed us more time to focus on improving the number and quality of features within the application.

Although the application can be run in single-user mode, it was designed from the start to support multiple simultaneous users interacting over a network. For this reason, all development and testing was performed using multiple networked machines. Features were added (geometry loading, user/roller interaction, paving, etc.) sequentially, incorporating the network elements of each before moving to the next. In order to ensure that we were targeting a wide range of computers as platforms for the application, we employed one computer with a newly released high-performance graphics card and one with a much less capable graphics card from the late nineties. Although the CPUs on these machines were also not equally matched, our development experience demonstrated that the graphics card performance was the most significant factor impacting the performance of the application on both platforms. This performance differential proved to be a challenge, impacting both the display and manipulation of local geometry as well as cross-network synchronization. The solutions developed to overcome this limitation will be addressed in detail below in the detailed implementation discussion.

Figure 1 illustrates the logical structure of the primary simulation application. The server component is discussed in the network implementation section below.

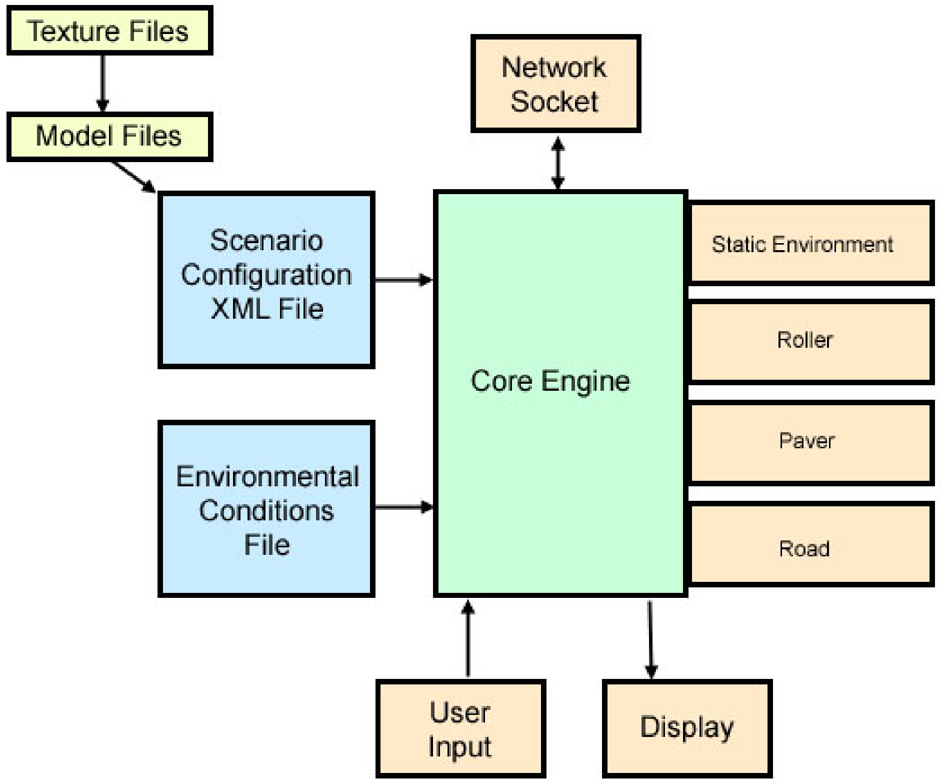


Figure 1. Application architecture

4.2 Scenario Implementation

We use the term scenario to describe the set of configurable elements that make up a simulation. This includes visual elements such as the geometry that makes up the setting in which the simulation takes place, functional elements such as the number of roller passes that will result in optimal road compaction, and elements that have both a visual and functional impact, such as the width of a roller. One of the primary goals of this project was to allow a wide range of scenarios to be executed with a single application.
To do this, we used a design that has been embraced by the video game industry which strives to separate the core engine of the game, which is hard-coded within the application, from the data, which is stored in a separate file and loaded at runtime. This design has several advantages. The first advantage is that the game or simulation engine is stripped down to its essential components resulting in cleaner code that is easier to modify and optimize. The second advantage is that the code does not need to be edited or recompiled in order to modify the simulation and, depending on the format used to store the data file, programming may not be required. In the game industry, this allows artists and designers to create new game content. In the case of this simulation application, new content could be added by simulation administrators or road construction experts.

4.2.1 The Scenario XML File Format

For many applications, the data file format used to drive the engine is stored in a binary format in order to speed loading times and obfuscate the content to consumers. For the paving simulation, we decided that an open, human-readable format would be most appropriate. We decided to store scenario information using XML. This format will require some knowledge of XML for any scenario authors, but our format is relatively simple. The self-describing and platform-neutral nature of XML would also ease the development of a scenario editing application, if the demand for one arose.

All customizable elements of a scenario are stored in an XML file except for the environmental conditions which can be selected at runtime and can be used with any scenario. The environmental conditions are defined in files generated by the program Multicool **(reference)**. At startup, in both single and multi-user modes, the user will be prompted to choose from a dynamically generated list of the files with the .xml extension stored in the Scenarios directory beneath the application directory. New scenarios can be added by simply creating a new scenario XML file and placing it in this directory.

Throughout this discussion of the customizable scenario feature of the simulation application, XML snippets will be provided to illustrate the format that should be used to describe each component. In addition, the sample scenarios provided with the application can be useful in illustrating the file format.

The root element of all scenario files should be a **scenario** element. All other data should be contained in child elements of this node.

```
<scenario>
      <!-- Scenario data is contained within this element -->
</scenario>
```

For example, the **description** element is used by the application when generating the menu of available scenarios. Because the description is displayed on a single line, it will need to be fairly brief.

```
<scenario>
      <description>A single lane road with 2 rollers</description>
      <!—Followed by additional scenario data -->
</scenario>
```

The **description** element is optional. If a scenario file does not contain one, the menu item will read “Description not available”. The discussion that follows will explain which elements are optional and which are required.

A Note about Environmental Conditions Files

The Multicool application allows the user to set parameter values for ambient temperature, windspeed, surface conditions, and pavement specifications, and generates a file listing the change of mat temperature with time. As it does with scenario files, the simulation application allows the user to choose the environmental conditions at runtime by dynamically generating a menu containing a list of the files contained in the "Multicool Files" directory below the application directory. These files should use the Multicool default .xls extension and, because no description is contained in the Multicool file format, the filename is used in the menu and so they should be named in a descriptive manner such as "SunnyWarm.xls" or "ColdRainy.xls".

4.2.2 The Static Environment

The static environment includes all of the visual elements of a simulation that are not interactive. Referred to in the code, as well as in the scenario file, as the "world", the static environment helps to make a scenario rich and immersive.

Figures 2 and 3 show the scenarios with two different static environments. The paver seen in both images is not part of the static environment.

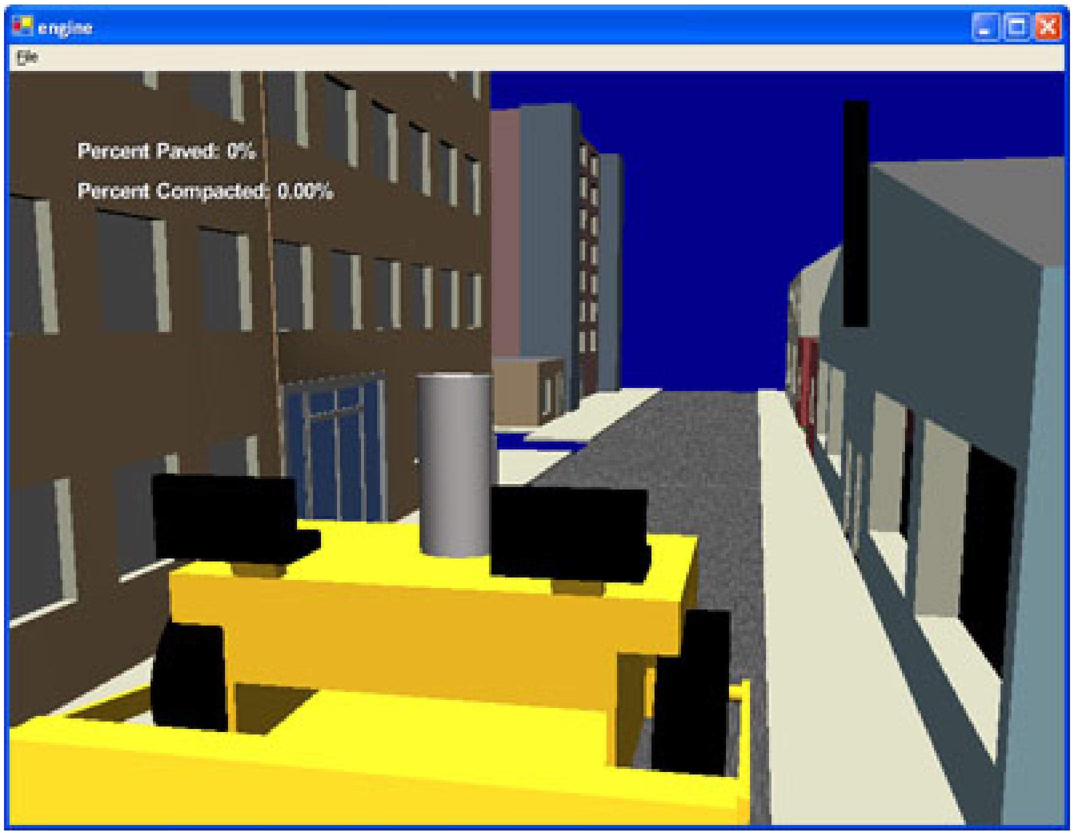


Figure 2. City environment

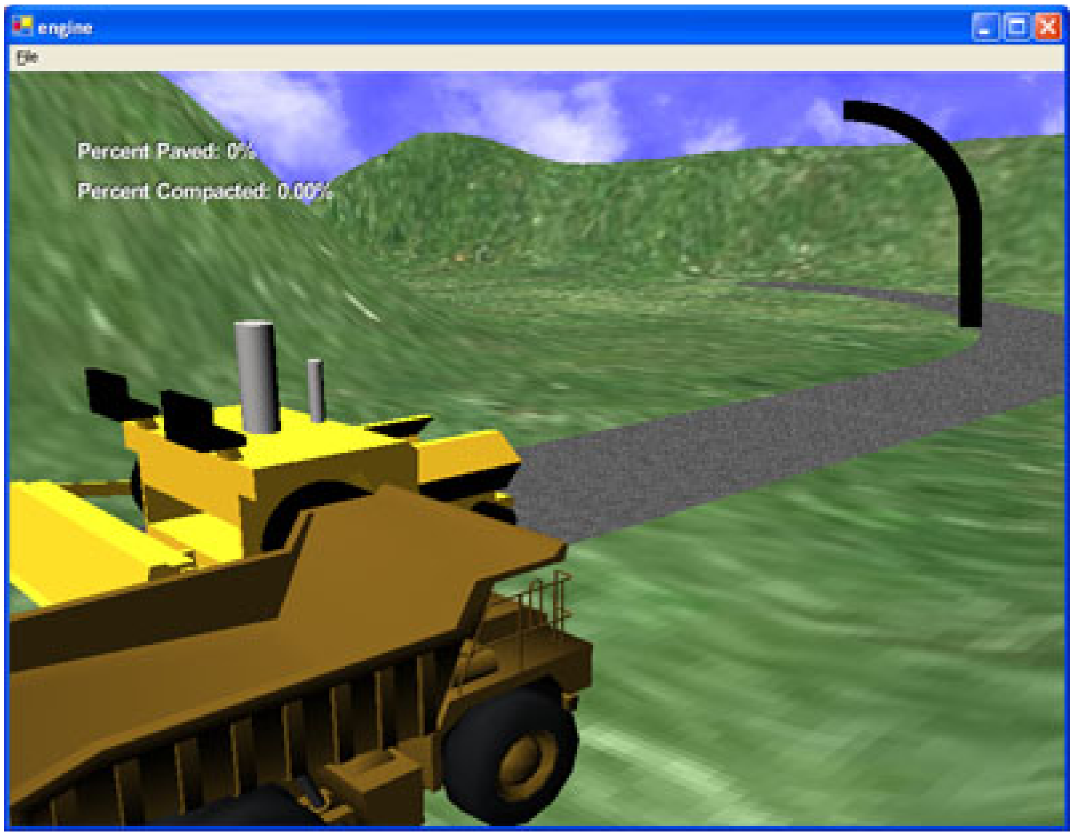


Figure 3. Curved two-lane road

The static environment is described in the scenario file by using multiple **mesh** elements embedded within a single **world** element. The **world** element is required, but may be empty if no static geometry is desired.

Each **mesh** element is used to reference a mesh file that is in the DirectX mesh file format (referred to as an X File for its .x extension). A wide array of models in this format can be found in commercial libraries and for free on the Internet. The dumptruck at the bottom of the above screen shot was obtained from a commercial library. There also exist a large number of converters for generating these files from other 3D graphics file formats.[1]

Table 1 describes the attributes of the **mesh** element used to build static environments.

Attribute	**Required**	**Description**
id	Yes	A unique identifier for the mesh object. This must be a different value for each **mesh** element.
meshFileName	Yes	The .x file containing the mesh data. This file must be present in the "Models" directory below the application directory. If any

[1] It should be noted that the simulation application displays non-static geometry in units of inches. This means that to create a cube mesh that, when compared to the paver, roller, and road, appears to be 2 feet tall, the vertices describing its geometry should be 24 units apart.

		textures are used, the .x file should address them without using a path. The application will look for all texture files in the “Textures” directory below the application directory.
tx	No	The translation of the object along the X axis.
ty	No	The translation of the object along the Y axis.
tz	No	The translation of the object along the Z axis.
ry	No	The local yaw rotation of the object (applied before translation).
rp	No	The local pitch rotation of the object (applied before translation).
rr	No	The local roll rotation of the object (applied before translation).
scale	No	The amount of scaling applied to the object. E.g. A value of 1 will display the object with no scaling. A value of 2 will make the object display twice its original size.
unlit	No	If this value is “true”, no lighting calculations will be performed on the object. It will display as if the ambient light in the scene is set at 100%.

Table 1. Mesh XML attributes

The following XML snippets show the **world** elements for the two screenshots included in this section. Note that the world containing the city block of buildings uses a single mesh create the static environment.

```
<world>
      <!-- world element describing the city block-->
      <mesh
            id="block1"
            meshFileName="block.x"
            tx="240"
            scale="1"
            unlit="false">
      </mesh>

      <mesh
            id="block2"
            meshFileName="block.x"
            tx="-2400"
            scale="1"
            unlit="false">
      </mesh>

</world>
```

```
<world>
      <!--world with hilly ground, sky, and dump truck-->
      <mesh
            id="ground"
            meshFileName="hill.x"
            ty="790"
            tx="-1400"
            tz="770"
            ry="-1.57"
            scale="550"
            unlit="true">
      </mesh>
      <mesh
            id="sky"
            meshFileName="skysphere.x"
            ty="-1800"
            scale="20"
            unlit="true">
      </mesh>
      <mesh
            id="dump1"
            meshFileName="dumptruck.x"
            ry="1.57"
            tx="400"
            scale="2">
      </mesh>
</world>
```

4.2.3 The Road

Even though the road in the simulation does not move, it is treated separately from the static environment. This is because the road is active, maintaining the state of the simulation. Each road is made up of one or more road segments which are treated both functionally and geometrically as a 2-dimensional array of points. These points are used to render the road onto the screen. Each point also has a state of being paved or unpaved, a temperature, and a number of passes that a roller has made over it after it has been paved.

Within the application code, as each frame is drawn, the current and previous position of the roller is passed to the road and it determines which, if any, points on the road have been passed over and updates its state as necessary. As the compacted state of each point is updated, it modifies its color accordingly.

The road also determines the path of the paver, which is defined as a path down the center of each road segment. If the paver is active, the core engine queries the road for the new position of the roller every frame. Every time the paver position passes over a row of points, that row is set to the paved state, indicating that roller passes will now influence each of the points' compacted state.

The road also manages its own HUD (heads up display) temperature readout in the upper right hand corner of the application window. Every minute, the core engine notifies the road that a minute has passed at which time the road updates the temperature value of each paved point in the road and changes the colors that are displayed in the HUD to indicate the updated temperature.

In multi-user simulations, each of the networked application roads' behavior is modified in order to optimize the synchronization between clients. This is discussed in the network implementation section.

Roads are defined in the scenario file with the **road** element. Only a single **road** element can be present in each scenario. Within this element are two required attributes, **optimalPasses** and **maximumPasses**. The optimal passes attribute indicates the number of passes with a roller that is desired for the best compaction given the parameters of the scenario. When a point on the road has not been compacted at all, it will be displayed as red. When a point on the road is optimally compacted, it will be displayed as green. Passes between none and optimal are shaded a color proportionally between the two. As a point on the road is overcompacted, its color is shifted toward blue until it reaches the maximum number of passes when it is set to purple and is no longer modified by roller activity.

```
<road
      optimalPasses="3"
      maximumPasses="4"
>
      <!-- road segments-->
</road>
```

4.2.4 Road Segments

A goal of this project was to allow scenarios to use different shapes and sizes of roads. We decided on a design that would allow great flexibility in road layouts without making the design and definition of the geometry too complex. The road in each simulation is defined as a set of one or more road segments. Each segment can be either straight or curved. By using combinations these two road segment types, more complex geometries can be created. Figure 4 shows a single straight road segment represented as a quad and the parameters that are used to define it.

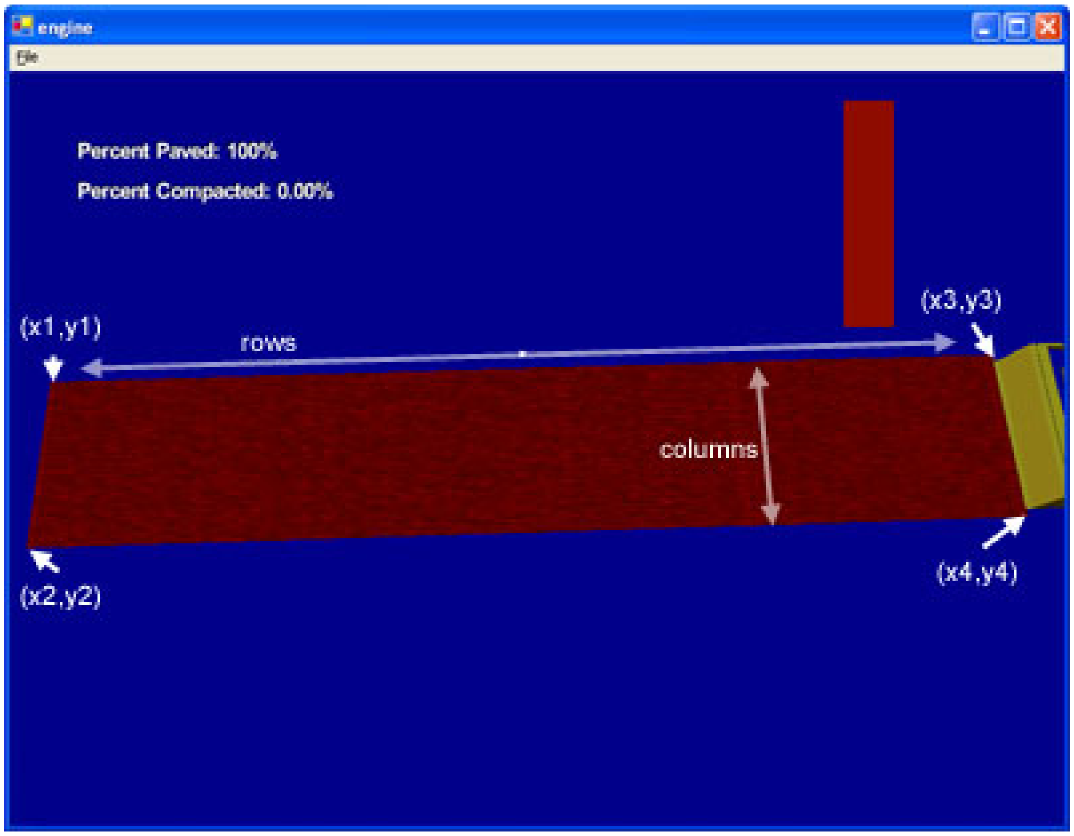


Figure 4. Quad road segment parameters

The x and y parameters define the four corners of the quad road segment. They do not have to be orthogonal or define a rectangle. The rows and columns parameters do not modify the size or shape of the road segment, but determine the number of subdivisions into which the road segment will be divided when the two dimensional array representing it is generated. Using a higher number for these values will result in a higher resolution when determining which points in the road a roller has passed over. Setting these values higher will also result in more triangles being drawn for each frame and therefore may bog down less capable video cards.[2]

The following snippet shows the definition for the road pictured above. All **roadSegment** elements must reside within a single **road** element in each scenario file. Remember that objects in the simulation application are represented as real dimensions in inches, so the road below is defined to be 20 feet wide and 100 feet long.

```
<road
      optimalPasses="3"
      maximumPasses="4"
>
      <roadSegment
            type="quad"
            x1="0.0"
            y1="0.0"
```

[2] Some older video cards have an upper limit of two bytes on index buffers, limiting the number of triangles that can be contained in a single object. If a very long road or a very high density of subdivisions is needed, it is recommended that multiple smaller road segments be used instead of one large one.

```
            x2="240.0"
            y2="0.0"
            x3="0.0"
            y3="1200.0"
            x4="240.0"
            y4="1200.0"
            rows="40"
            columns="10">
      </roadSegment>
</road>
```

Curved road segments are stored internally in the same way as quadrilateral straight segments, but they are defined differently. Figure 5 shows a curved road segment and the definition parameters.

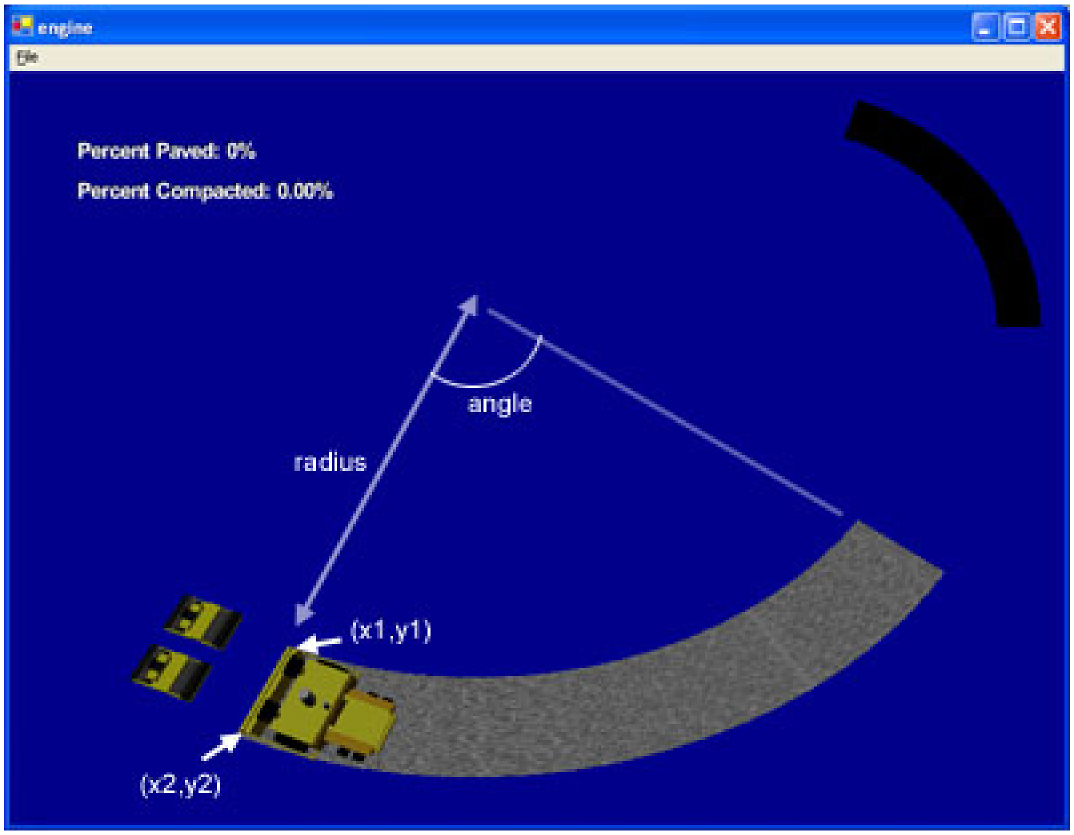


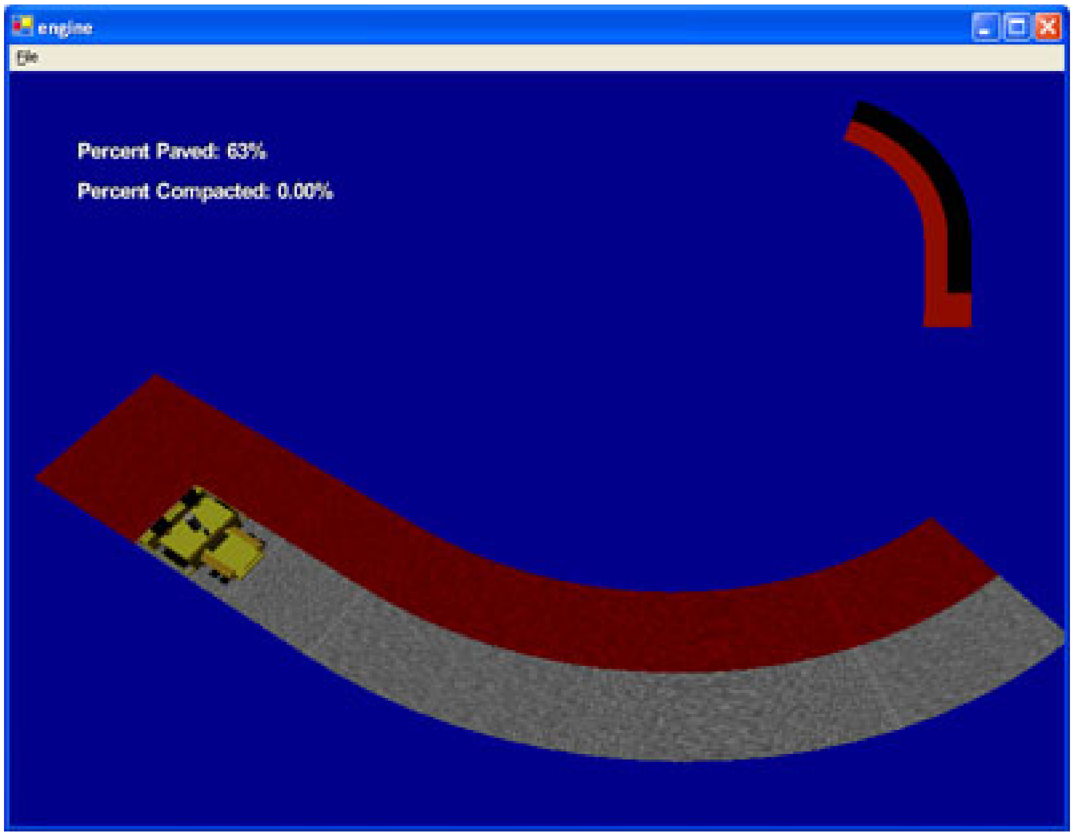
Figure 5. Curved segment parameters

The x and y parameters indicate the starting points of the road. This line is swept along a circular path, the radius of which is set by the **radius** parameter. The angle, which is given in radians determines the length of the arc swept out by the road segment. To make the road curve to the right instead of to the left, a negative value should be provided for the **radius** parameter. The **rows** and **columns** definitions function the same with curved road segments as they do with quads. The following XML shows a curved road segment definition.

```
<road
      optimalPasses="3"
      maximumPasses="4"
```

```
>
      <roadSegment
            type="curve"
            x1="0.0"
            y1="0.0"
            x2="240.0"
            y2="0.0"
            radius="1200"
            angle="1.25"
            rows="20"
            columns="10">
      </roadSegment>
</road>
```

The curved and straight road segments can be combined to create more complicated road configurations. They can also be used to create multi-lane roads. The paver follows the path down the middle of the road segments in the order in which they are defined in the file. Defining two road segments next to each other will create a multilane road that the paver will pave in two passes. Figure 6 shows a two lane road composed of 4 straight and curved road segments.



Percent Paved: 63%

Percent Compacted: 0.00%

Figure 6. Multi-segment road

The following XML from the scenario file defines this road.

```
<road
      optimalPasses="3"
      maximumPasses="4"
>
```

```
      <roadSegment
            type="quad"
            x1="0.0"
            y1="-1000.0"
            x2="240.0"
            y2="-1000.0"
            x3="0.0"
            y3="0.0"
            x4="240.0"
            y4="0.0"
            rows="40"
            columns="10">
      </roadSegment>
      <roadSegment
            type="curve"
            x1="0.0"
            y1="0.0"
            x2="240.0"
            y2="0.0"
            radius="1200"
            angle="1.25"
            rows="20"
            columns="10">
      </roadSegment>
      <roadSegment
            type="quad"
            x1="240.0"
            y1="-1000.0"
            x2="480.0"
            y2="-1000.0"
            x3="240.0"
            y3="0.0"
            x4="480.0"
            y4="0.0"
            rows="40"
            columns="10">
      </roadSegment>
      <roadSegment
            type="curve"
            x1="240.0"
            y1="0.0"
            x2="480.0"
            y2="0.0"
            radius="1440"
            angle="1.25"
            rows="20"
            columns="10">
      </roadSegment>
</road>
```

4.2.5 The Roller

The simulation defines a single roller class and creates an instance of this class for each roller in a scenario. Each roller instance maintains its own state and position and exposes an interface to the core engine similar to the interface of a real world roller. It has methods for moving forward and backward and turning right and left, as well as braking. As each frame is drawn, the roller is queried by the core engine for the new and previous locations of its drum. This data is passed to the road, which determines if any paved area of the road has been passed over.

One or more **rollerInstance** elements should be placed in the scenario file to define the rollers for the scenario. This element contains the **initialRollerX** and **initialRollerZ** attributes to define the position in the world where the roller will be placed when the simulation begins. The **initialRollerRotation** attribute is a value, in radians, that describes the initial orientation of the roller. The optional **rollerScale** attribute is used to adjust the width of the roller. If this value is omitted or set to 1, the roller will be the default width, which is 7 feet (this is the width of the drum). A value of 2 will set the roller to be 14 feet wide, and a value of .5 will set the roller to be 3.5 feet wide. The width of the roller is not merely cosmetic and will affect the amount of the road that will be compacted with each pass.

The following example shows a roller instance definition from a scenario file.

```
<rollerInstance
      initialRollerRotation="-1.57"
      initialRollerX="160.0"
      initialRollerZ="-200.0"
      rollerScale="1.0">
</rollerInstance>
```

4.2.6 The Paver

Each scenario contains a single paver. Unlike the roller, the paver does not maintain its own state. As explained in the road section above, the road determines the path of the paver and determines when a row of points in the road is paved. The core engine queries the road each frame for the position and orientation of the paver and then calls a method on the paver to move it to this location.

The two configurable attributes of the paver in the scenario file are its speed and its size. The units of the paver speed is inches moved per frame. This assumes a 30 frames per second frame rate, so a value of 1 will set the speed at 30 inches per second. A value of .1 will set the speed to 3 inches per second. This rate is adjusted to accommodate varying frame rates as described in the network implementation section below. The default scale of the paver is 14 feet wide. Setting its scale value to .5 will make it 7 feet wide. Unlike the roller, the width of the paver only affects the appearance of the simulation. Regardless of the scale value used, one complete row of each road segment will be paved as the paver passes over. In order to have the paver pave only a portion of the width of a road, a multilane road should be used as described in the road section above.

The following example shows a paver definition from a scenario file.

```
<paver speed="5"
      paverScale="1">
</paver>
```

4.3 Network Implementation

The network component of the simulation application is the most complex, requiring more man hours to develop than any other feature and comprising the bulk of the application code. From the beginning, we recognized that the most critical constraint of the multi-user simulation was that synchronization between all clients must be maintained. A lack of synchronization would be most noticeable visually, for example if two users viewed each others rollers remotely in different positions than they appeared on their local application. More importantly, a lack of synchronization could affect the functional aspect of the simulation, for example if the simulation indicated that a section of road was optimally compacted to one user and under compacted to another.

A challenge to maintaining synchronization is the varying performance of the computers on which the simulation is run. This performance turned out to be primarily influenced by the capabilities of the graphics card on the machine. During our development process we used graphics cards on the far ends of the performance curve which highlighted this issue. The first signs of the performance issue appeared when we had completed the roller model and built the functionality allowing the user to drive it. Before even adding the networking code to allow the rollers to be driven remotely, we noticed that the roller on the slower computer drove significantly slower than that on the fast computer. This behavior was not unexpected as we had defined the rollers to move a specified number of units per frame. It follows that the application with the higher frame rate would have a roller that moved more quickly. Our solution to this issue was to create a variable which we called the time scalar. This variable was set using the following logic.

```
time = DXUtil.timeGetTime();
engineElapsedTime = (double) (time - lastTime);
timeScalar=30.0*engineElapsedTime*.001;
lastTime = time;
```

The **timeGetTime** method, provided by the DirectX utility library, returns the number of milliseconds since the application started and allows us to obtain the time span in milliseconds that it takes to draw a single frame. This value was then scaled, dividing by 1000 to get seconds and then multiplied by 30. The time scalar is then a ration of an ideal 30 frames per second (fps) rate to the application's current frame rate. The time scalar, calculated at each frame, is used to scale phenomena that need to transpire at the same rate on all clients.

For example, the roller could be assigned to move by adding 10 units to its translation along the X axis. If the application is running at 30 fps, the time scalar will be 1 and therefore the roller will move 10 units per frame. If the application is running on a faster machine at 45 fps, the time scalar would be .66 and the roller would move 6.6 units per frame. If the application is running on a slower machine at 15 fps, the time scalar would be 2 and the roller would move 20 units per frame. Each of these conditions results in the roller moving 300 units after one second.

The addition of the time scalar, though theoretically sound, was not a perfect solution. The precision of the system call returning the elapsed time was such that the time scalar value was not always sufficiently accurate. Large scale tests, such as driving the roller a set distance on both machines, proved fairly accurate – the rollers arrived at the goal in the same elapsed time.

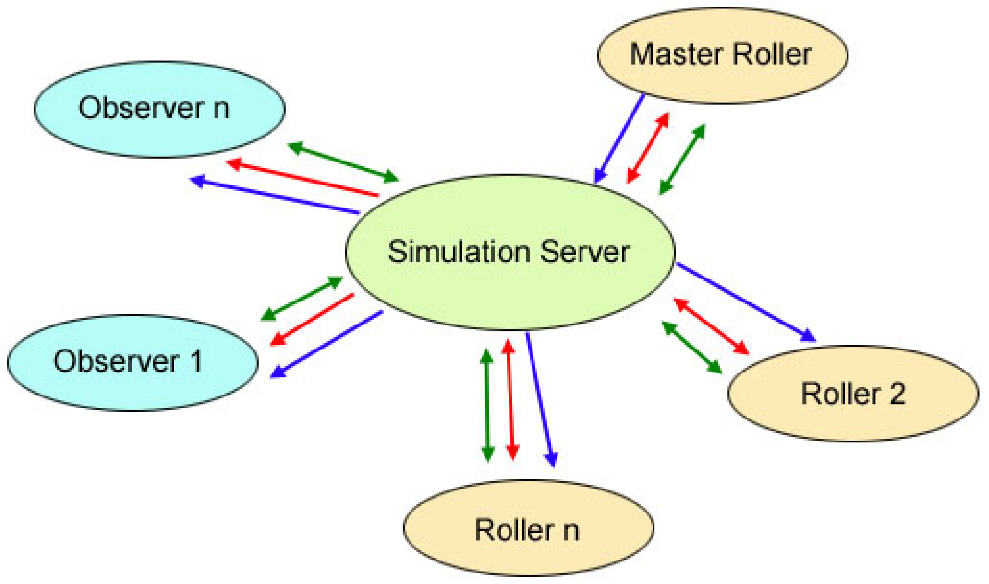
But the small fluctuations on the per frame scale resulted in some performance degradation and jerkiness as the cross-network features were added. When all client machines in a simulation have comparable hardware capabilities, this performance issue is not noticeable.

4.3.1 The Simulation Server

Initially, the network simulation architecture did not include a separate server component. Instead, the server functionality was built into the simulation application. In a scenario involving only two participants, this was preferred because one of the two clients would conduct transactions with the server through local method calls. This meant that the number of trips over the network that the data would take would be half of that required if both clients were sending data over the network to a server. As we considered expanding the multi-user simulation to allow n number of clients, it became clear that the computational overhead of the server-hosting application would become problematic for performance. Also, the implementation of new network features would become much more complex as each networked event would have to be implemented in both network protocol and local method calls. For this reason, we decided to make the simulation server a separate application.

The server design we used is fairly simple. The server maintains a list of sockets, each of which is connected to a client. For the duration of the server's execution, it cycles through this list, checking to see if any data is waiting to be read from each socket. If there is data, the server loops through all of the other sockets writes the data out. Then it goes back to checking each socket for data to be read. Except for some handshaking operations that are performed when the simulation is being launched and the clients are connecting, the server is completely agnostic of the data being passed through it or the role (roller or observer) of the clients connected to its sockets. It does not need to make any decisions as to how data should be routed. All data that is received is simply echoed out to all of the clients except for the one that sent it.

The following diagram illustrates the flow of data through the server.



🟩 Paver start/stop, text message. Sent and received by all clients.
🟥 Roller/Paver position, road compaction data. Sent and received by rollers. Received by observers.
🟦 Master paving data. Sent by master roller. Received by all other clients.

Figure 7. Data flow through the server

The three situations in which the server must evaluate the data being transmitted through it all occur during the startup of the simulation. First, the client that initiates the simulation will choose a scenario from the list of available files. The client passes the scenario file name to all other clients as they connect. The same thing is done with the environmental conditions file name selected by the initial client.

The server also tracks the number of clients that have connected and requested a roller role. When a new client connects, it sends this number to the client. The client compares this against the number of rollers in the scenario file and determines whether or not to allow the user to select the roller role. After this initial handshaking is completed, the server does not evaluate the data being sent between clients.

4.3.2 The Network Protocol

To send commands between client applications, a simple network protocol was established. This is a high-level protocol that sits on top of the TCP/IP layer. Every piece of data sent to the network by a client is referred to as a command. These commands take the form of a marker for the beginning of the command and a marker for the end. Between these two markers is a unique identifier indicating the command being transmitted followed by the data associated with the command. This command framework made it easy to add new commands as new features were added. For each new command, code is added to construct the command protocol for transmission and for deconstructing the command after it has been received.

4.3.3 Update Timing

Although it is miniscule compared to the time it takes to render the scene, the processing of the protocol commands does use some CPU time. For this reason we decided not to send or check for received network data every frame. Once again, the performance difference between graphics cards impacted this decision. For example, if all client applications send and receive data every three frames, then a client running at 15 fps will be overwhelmed by the deluge of data from a client running at 45 fps. Once again, we used the time scalar to schedule network updates in attempt to have all applications send and receive at the same interval, regardless of their frame rate.

This was largely successful, but not perfect because of the precision of the frame rate. This becomes an issue when the rate of network updates approaches the overall frame rate of the application on given client (which occurs on machines with slow CPUs and graphics cards). The result is that a roller controlled by a slower computer is displayed with some stuttering on a faster machine in the multi-user simulation. The same effect works the other direction, but because the frame rate on the slower machine is low, the stuttering is not as pronounced compared to the movement of the local roller.

4.3.4 Synchronizing the Rollers

As stated in the previous discussion, the time scalar helps to ensure that on a large scale, the rollers are moving at the same speed on all clients. However, the lack of accuracy on the per frame scale makes it impossible to guarantee that the functional operations of the roller can be kept in sync. For example, while an application will display a remote roller's current position precisely once it has stopped moving, there is no mechanism to ensure that the roller arrived at the position via the exact same path as it did on the remote machine, and therefore it cannot be assumed that the exact same points on the road are updated as being compacted.[3] For this reason, each client application that is acting as a roller in the simulation performs the test to see which points on the road its own roller passed over. It accumulates a list of these points and periodically transmits them to the server which in turn transmits the list to all of the other connected clients. This design has the added benefit of distributing the computational load of detecting compacted points among the connected clients.

4.3.5 Syncronizing the Paver

The same synchronization issue occurs with the paver. While the paver will be displayed in a basically synchronized manner across all clients, the accuracy of determining when a row of road has been paved cannot be trusted. For this reason, a single roller – called the master roller – is responsible for determining when a row is paved and broadcasting it to the other clients. The first client that connects to the server and requests a roller role is the master roller. However, all

[3] The scale of this phenomenon is on the order of a fraction of a second.

other clients are able to start and stop the paver by issuing a single network command that is only acted on by the master roller.

4.3.6 Synchronizing the Temperature

Once again, the master roller is responsible for keeping track of the pavement cooling for all connected clients. Each client loads the same environmental conditions file and has a list of the temperature associated with each minute that pavement has been laid down. The master roller essentially keeps global time for the simulation, sending out a command once a minute that all points on the road that are paved should decrement their temperature value to the next in the list.

5. Summary and Future Work

Virtual training tools provide a new way of presenting training materials and illustrating the process, constraints, and complexities of pavement construction. Using a game-like metaphor, simulations that are interactive and graphically rich provide a compelling mechanism to engage the user in the training process. Beyond the physical aspects of construction, the introduction of network-enabled multi-user simulations increase the value and realism of the training by incorporating the group dynamics of the construction process such as the cooperation and coordination of construction personnel. In this work we demonstrated, through a prototype system, that such training simulations can be developed to run on currently available consumer computers.

The development of the virtual paving simulation application highlighted the value of the cooperative, multi-user aspects of training simulations. Do to the limitations of computer interfaces, the ability to realistically mimic the mechanical aspects of performing micro-level paving tasks, such as driving a pavement compactor, is inherently limited. However, the inclusion of multiple participants in a macro-level simulation, focusing on cooperation and communication, has great potential to improve the efficiency of paving construction and the quality of the finished road.

Future work will concentrate on the high-level complexities of managing pavement construction with less focus on the physical completion of specific construction tasks and viewing the problem more in terms of resource management. This includes the both the management of construction personnel as well as materials and equipment. Future simulations should illustrate the importance of planning and timing of a broader range of paving tasks. Artificially intelligent automatons should be developed to execute the paving tasks, allowing the user to focus on timing and coordination of tasks rather than the actual physical execution.

Appendix A. Walkthrough and User's manual

This section walks through the user experience of the virtual hotmix paving simulation, discussing the application's features and controls.

Starting a New Simulation

A user launches the paving simulator by double clicking the program icon. After the application loads, the user is presented with a menu that will determine what type of simulation will be run.

Single User Simulation

If a single user simulation is selected, the user will first be prompted to select a paving scenario from a list. This list is populated from the set of files in the Scenarios directory under the application directory. Scenario files describe the world in which the simulation takes place as well as configures the functional parameters of the situation. For example, the shape of the road to be paved can be modified as well as the number of compactor passes considered optimal for the simulation. Scenarios files can be created and modified in order to customize the paving simulation. For information on the scenario file format, see the implementation section.

After choosing a paving scenario, the user will be prompted to choose an environmental conditions file that will determine the speed with which the hotmix asphalt will cool, and therefore the time the user has to completely compact the road.

Finally, the user will be prompted to hit any key to begin the simulation.

Start New Network Simulation

If the user chooses to start a new network simulation, the user can choose to have the simulation application automatically launch the server or the server can be launched manually by double clicking the server icon either on the same computer that the simulation application is running on, or on another machine.

The user will then be prompted to enter the IP address of the computer on which the server is running. This address allows the application to connect to the server.

Note on IP addresses and firewalls: The IP address of a Windows computer can be determined by opening a DOS command prompt and using the "ipconfig" command. However, sometimes the IP address returned by this command will not be accessible from a computer on another network. Also, today many computers run firewall applications that prevent network access to its network ports. The simulation server and client applications use port 9000 to communicate. This port will need to be accessible between clients for the simulation to run.

After the IP Address has been entered, the user will be prompted to make sure the server is running and then hit any key to continue. The server status indicator will read "Listening on port 9000 on xxx.xxx.xxx.xxx" when it is running.

Next the user will be prompted to choose a scenario and then an environmental file for the network simulation. The application will then display the list of roles that the user can take on. This role is either that of a roller operator or of an observer. The number of roller operators is determined by the scenario file that was chosen when the simulation was launched. After clients have joined and taken on all of the available roller operator roles, only the observer role will be available from the menu.

Finally, the user will be prompted for a name that will be displayed when sending text messages to other networked simulation users. After this, the user can hit any key to begin the simulation.

Join Existing Network Simulation

If the user chooses to join an existing network simulation, the application will request the IP address of the computer on which the server is being run. Before attempting to connect to the server, one should make sure that the server is running and that another user has already started a network simulation by choosing the "Start New Network Simulation" option. After connecting to the server, the application will load the scenario and environmental files for the simulation. The user will then be asked to select to take on either a roller operator role or that of an observer.

Finally, the user will be prompted to provide a name for text messaging. After this, the user can hit any key to join the simulation.

Viewing the world

The virtual paving simulation provides a flexible viewing mechanism allowing simulation participants to observe the world and the paving process from any angle.

When the simulation begins, users who have selected the roller operator role (this role is the default in single user simulations) will see the world from behind and slightly above the roller that they control. Users who joined the simulation as observers will see a view from above, looking down on the simulation.

The view can be switched between the roller-centric view and the observer view by hitting the C key.

Each of these views can be moved by using Q, W, A, Z, S and X keys. The Q and W keys move the view left and right and the A and Z keys move the view up and down, relative to the user's line of sight. The S key moves the view forward in the direction that the user is looking, while the X key moves the view back.

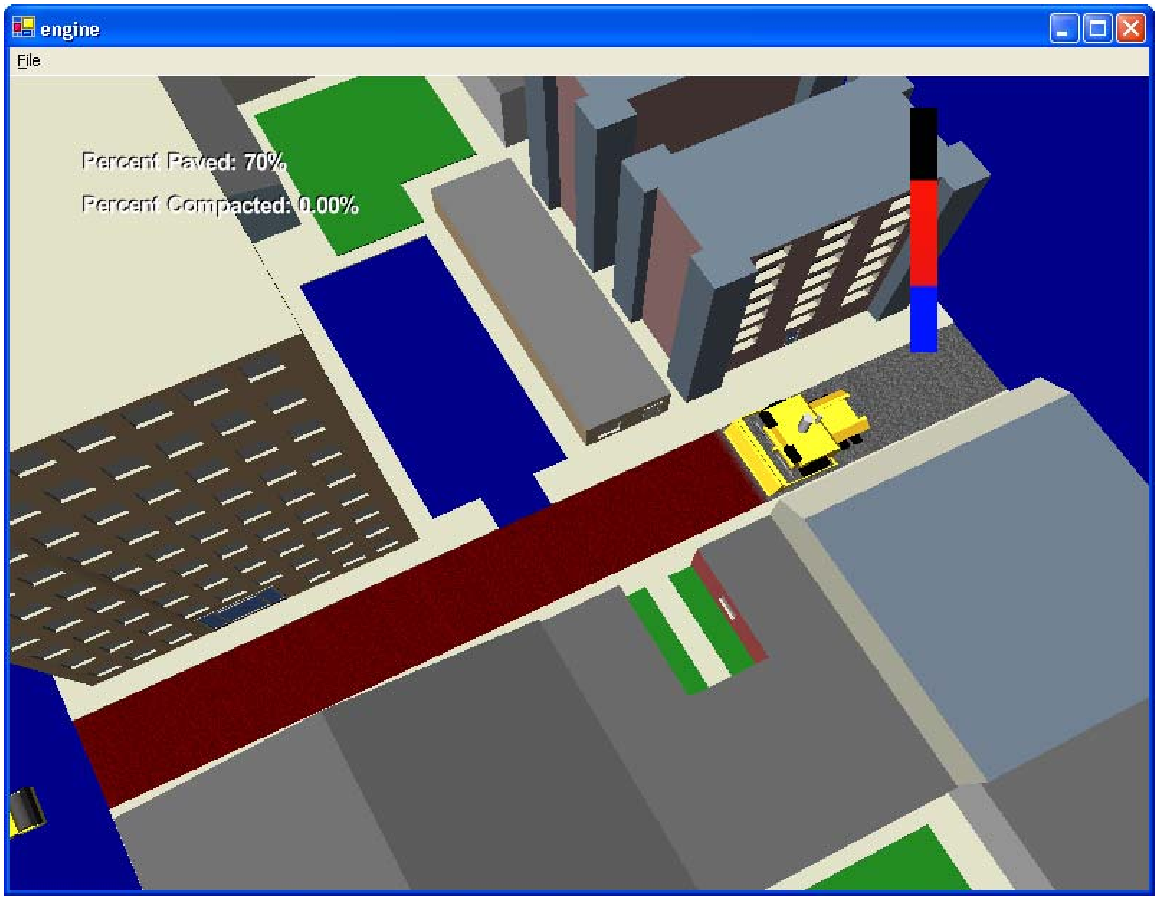
The view that started behind the roller is always relative to the roller. This means that however the view is adjusted, as the roller moves, so will the user's view. The observer view is not relative to the roller. It remains stationary regardless of the roller's movement.

Running the Paver

Before the road can be compacted, the hotmix must be laid by the paver. The paver begins the scenario at the beginning of the road. Any participant in the simulation, including observers, may start the paver by pressing the P key. The paver automatically drives along the road, paving at the rate defined in the selected scenario file. The paver can be stopped by pressing the P key again. This allows the users to pave a section of the road, compact that section, and then start the paver again. The percentage of the total road in the simulation that has been paved is displayed in the upper left hand corner of the window.

The temperature gauge is displayed as a map of the road in the upper right-hand corner of the window. This map indicates the temperature of the hotmix at all points on the road. The gauge will take the same shape as the road being paved. As the road is paved, the temperature gauge will turn red as hotmix is laid down. As the simulation continues and the pavement cools, the temperature gauge will change color. When the pavement has cooled completely, as indicated by the environmental conditions file chosen for the simulation, the gauge will be blue.

The following screen shot illustrates the temperature gauge functionality. The gauge at the right indicates that the beginning section of the road was paved several minutes previously and has almost entirely cooled. The middle section of the road has just been paved and is hot. The final section of the road has not been paved.



Percent Paved: 70%

Percent Compacted: 0.00%

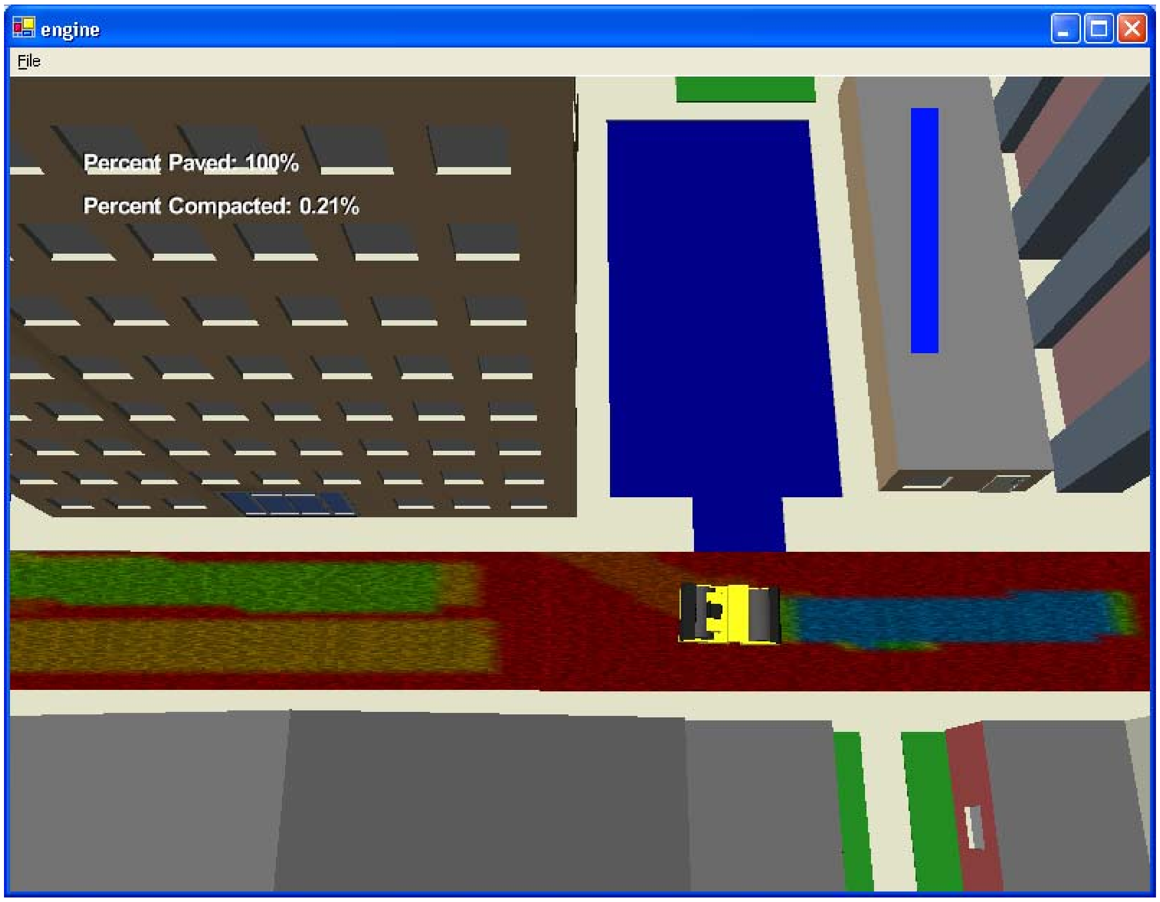
Driving the Roller

The roller is driven by using the arrow keys. The left and right arrow keys steer the roller left and right respectively. Holding the up and down arrow keys drives the roller forward and backward.

The color of the road indicates to the user the number of passes the roller has made over any point on the road. Before the road is paved, it will be grey in color. As the paver lays the hotmix, the color will change to red, indicating that the pavement has not been compacted at all. The color of the road will change as the rollers pass over it, shifting from red to orange and then yellow. When the optimal number of passes, as determined by the scenario file, is reached, the road will be green. All rollers in a multi-user simulation will contribute to the total number of passes for the road. As the road becomes overcompacted, the color will shift toward blue and finally become purple, indicating that the maximum number of passes allowed by the simulation has been reached.

As the road is compacted, the percent of total compaction is displayed in the upper left hand corner of the window.

The following screen shot shows the road color that indicates the number of roller passes on the road. The optimal number of passes for this scenario is 4. The section of road on top on the left has been passed over 4 times. The section on the on the bottom on the left is under compacted. It has been passed over 2 times. The section on the right has been over compacted with 6 passes. The rest of the road is red, indicating that it has not been compacted at all.



Text Messaging

The text messaging feature of the simulation is designed to allow participants in the simulation to coordinate their actions. It can also be used by the simulation administrator to offer instructions to users that are driving rollers. Text messaging is only available during multi-user simulations.

To begin a text message, press the T key or click the “Talk” button on the screen. This will display a prompt at the bottom of the screen. While you are typing a message, none of the other simulation controls will function. Hitting the Enter key will send the text message, displaying it at the bottom of the screen for all simulation participants.