Proceedings of the 2003 Winter Simulation Conference
S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.

VARIABLE-SPEED RESOURCE MOTION IN ANIMATIONS OF DISCRETE-EVENT PROCESS MODELS

Vineet R. Kamat

Department of Civil and Environmental Engineering
University of Michigan
Ann Arbor, MI 48109-2125, U.S.A.

Julio C. Martinez

Department of Civil and Environmental Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061-0105, U.S.A.

ABSTRACT

This paper presents research that addresses the problem of describing the accurate, variable-speed motion of simulation objects on realistically-shaped trajectories (i.e. paths) in animations of discrete-event simulation models. The work puts in place techniques that modelers can use to instruct virtual simulation objects to follow any arbitrarily-shaped velocity profiles while adhering to fixed motion completion times when traversing along any defined motion path trajectories. A computation scheme that allows simulation models to define the general shapes of relevant velocity profiles and then heuristically scales those profiles to accommodate communicated activity instance durations is presented. While allowing animated simulation objects to be moved with any arbitrarily shaped velocity profiles, this technique ensures that an object's temporo-spatial control rests entirely with the underlying simulation models.

1 INTRODUCTION

Visualization of modeled operations can be of significant help in the verification and validation of discrete-event simulation (DES) models (Law and Kelton 2000). This is especially true in construction where typical decision makers are experts in their domain but are not generally proficient in simulation itself. Visualization can also provide decision makers with valuable insight into subtleties of planned construction operations that are otherwise non-quantifiable and non-presentable.

The authors' recent research efforts have focused on designing automatic, simulation-driven methods to visualize modeled construction processes and any evolving products in smooth, continuous, dynamic 3D virtual worlds. Methods have been designed to describe animated 3D worlds that show how simulated processes are carried out, using simple parametric text statements and references to 3D CAD models (Kamat and Martinez 2003a). This simple text animation description language, called VITASCOPE, is meant to be written out by end-user programmable tools such as DES systems and allows a computer to create a dynamic 3D virtual world that shows people, machines, and/or materials interacting as they perform the modeled processes.

1.1 Research Motivation

Synthetic, process simulation-driven 3D virtual worlds are spatially and temporally faithful to the underlying DES models that author the visualizations. Notwithstanding, the 3D visual representations (i.e. visualizations) of several modeled processes digress in time and space accuracy from the corresponding real-world operations due to the inherent characteristics of DES.

In DES, the state of a running model changes only at discrete, but possibly random sets of simulated time points (Schriber and Brunner 2001). These time points are typically the start or end of the model's activities, and it is only then that a running DES model can communicate with other processes, or perform other actions such as output to an animation trace. A DES model is only concerned about the time instants at which instances of modeled activities begin or end, and chooses to ignore everything (e.g. rate of activity performance) that happens in between. The information that a running DES model can communicate to external 3D animation methods is thus limited to the start times and durations of all activity instances that occur in any simulation run. The animation methods must then use these limited pieces of discrete information to generate a smooth, continuous, dynamic 3D virtual world representation that depicts the modeled activities being performed.

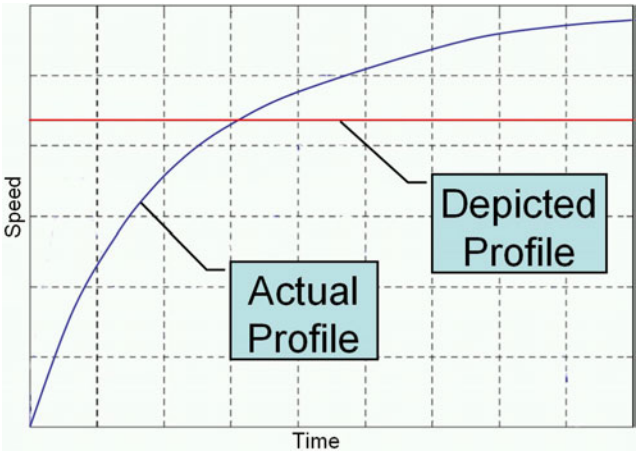
In any modeled activity that involves motion of simulation entities (e.g. a hauling truck, an airplane taking off), the only kinetic property that can be computed from the pieces of communicated activity instance information (i.e. start time and duration) is the average speed of the simulation entity (e.g. truck, airplane) in that particular activity instance (e.g. haul dirt, take off). This is precisely the computation existing 3D animation methods perform in describing the smooth, continuous motion of simulation entities in virtual worlds.

In particular, the motion of a simulation entity is depicted by transforming (i.e. moving) the pertinent instantiated CAD model of the entity at the computed average velocity. The simulation entity-representing CAD model is smoothly and linearly interpolated on a 3D trajectory that represents that entity's motion path (e.g. haul road, runway) in the communicated activity instance. Thus, a virtual airplane taking off appears to travel down a runway at a constant (average) speed for the activity instance's sampled time duration (i.e. runway occupancy time). Similarly, a loaded truck hauling dirt on a virtual earthmoving jobsite travels at a constant speed for the entire duration of the haul regardless of the grades on the 3D motion trajectory (i.e. virtual haul road) it travels on.

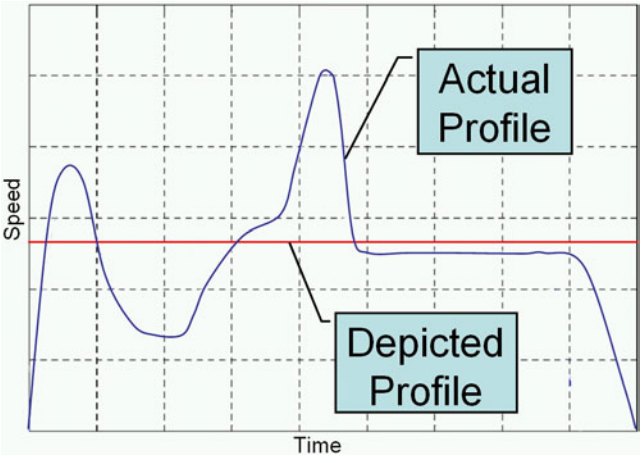
The temporal and spatial accuracy of simulation model-generated dynamic 3D virtual worlds is commensurate with the detail of the communicated information from which the visualizations are created. The existing animation scheme is faithful to DES in that animated activity instances inside a 3D virtual world begin and end at the exact time instants dictated by the visualization-authoring simulation models, with smooth, continuous, "constant-speed" intermediate motion of the involved virtual entities (i.e. simulation objects). Such visualization of simulation objects performing the modeled (and communicated) tasks at constant speed is often sufficient to verify and validate several DES models (Kamat and Martinez 2003b).

However, the depicted constant velocity profiles of moving virtual simulation objects are not an accurate representation of reality. For instance, an airplane taking off on a runway obviously does not travel at a constant velocity. Instead, as figure 1(a) presents, the airplane continuously accelerates as it races down a runway and takes-off. Similarly, as figure 1(b) presents, the typical velocity profile of a truck that hauls dirt is a function of several disparate factors such as engine power, load being hauled, and the rolling resistance and grade of haul road segments. Due to the limited operational information available from underlying DES models, existing methods of animating simulated processes are, however, unable to adopt such realistic velocity profiles in describing the motion of virtual simulation entities. Instead, commensurate with the available pieces of information, simulation objects are assumed to be moving at constant, average speeds with straight-line velocity profiles such as those superimposed on figures 1(a) and 1(b).

This assumption (and portrayal) of constant-speed entity motion can often hinder the validation of modeled processes in cases where the relative segmental speeds of moving simulation entities and/or their acceleration/deceleration influence their evolution and inter-object interactions in a modeled and visualized system (e.g. airport, earthmoving jobsite). In addition, such constant-speed visualization of modeled processes can frequently fail to elicit credibility for simulation models, especially from domain experts and decision makers who are not familiar with the mechanics of DES and/or are skeptic about simulation analyses beforehand.



(a) Airplane on Take-Off Roll



(b) Dumptruck on Haul Road

Figure 1: Typical Velocity Profiles of Real Objects

2 DESCRIBING VARIABLE SPEED MOTION

The only temporal information about an activity-instance that a running DES model can communicate to external processes such as 3D animation methods is that instance's start time and its duration. Due to inherent modeling features, a DES model can provide no information on the rate at which the task(s) in any activity instance are performed. This is unlike Continuous Simulation, where the state of a model (and hence the rate of activity performance) is continuously monitored at every time instant using differential equations of motion (Law and Kelton 2000). The performance rate of an activity, however, provides precisely the temporal information needed to describe the velocity profile(s) of simulation object(s) that move (i.e. travel) while performing the task(s) in a particular activity instance.

2.1 Hypothetical Hybrid Animation Approach

In order to describe variable-speed motion of virtual DES objects then, the first possible technique that was explored was to describe a parallel continuous simulation system that would tightly integrate with the methods of animating DES models in 3D. In particular, the authors considered the possibility of externally formulating an in-context simulation object's pertinent kinetic properties (e.g. nominal acceleration/deceleration, maximum permitted velocity) and using that information along with each communicated activity instance (i.e. start time and duration). During visualization, the integrated continuous simulation mechanism would use the formulated kinetic properties to compute the involved simulation object's velocity profile (i.e. its temporal evolution) beginning at the indicated activity instance start time.

This hybrid animation approach is, however, impossible to achieve in a DES framework. In particular, such a strategy would only work if a DES model communicated an activity instance's start time and enforced no restrictions on when it ended. That information (i.e. activity instance end time) could then be determined in real-time as the involved simulation object's temporal evolution was computed continuously during visualization. A DES model must, however, explicitly enforce a communicated activity instance's end time (i.e. its duration). This is obvious because the start of instance(s) of other simulation model activities (often involving the object that is in context in the current activity instance) is explicitly tied to the completion of the current communicated activity instance. Once instantiated, any instance of such a successive activity will attempt to exclusively manipulate the in-context virtual simulation object to visually describe the performance of that latter communicated task.

The hypothetical continuous simulation system computing the simulation object's temporal evolution in the instance of the previous activity would, however, be unable to guarantee the completion of its motion at the exact precise instant at which the successive activity starts. Stated differently, it is mathematically impossible to externally formulate kinetic object properties and compute a unique, valid, continuous velocity profile using a set of differential equations if the motion start and end times (i.e. the lower and upper bound of the integration interval) and the distance traversed (i.e. the area under the resulting curve) are both explicitly enforced.

This is however the case in DES. As such, any computation scheme (for visualizing simulated processes) that wrests the temporal and spatial control of simulation objects away from the underlying DES models cannot portray the modeled operations correctly in dynamic 3D virtual worlds. The description of any arbitrary velocity profiles to be applied to mobile simulation objects during animation must thus be sought from the DES models that author visualizations. However, a DES model obviously does not encapsulate any such information (e.g. an object's kinetic properties) simply because the rate of performing any modeled activity is generally irrelevant to the model from the simulation analysis perspective.

2.2 Time-Based Scaling of Velocity Profile Shapes

In order to visually describe variable speed motion of animated simulation objects then, the authors devised a unique computation scheme that prudently shares the responsibility of describing a moving simulation object's arbitrary velocity profile between the underlying DES model and the 3D animation methods. In particular, the general shape of the velocity profile to be applied to a moving object is sought from the visualization-authoring DES model. The shape of this general profile can be explicitly defined (and input into a DES model) by a modeler, or it can be the result of computations performed within a running model.

Then, at each communicated activity instance, the animation methods heuristically scale (up or down) the previously defined velocity profile in such a way that the mobile simulation object to which it is applied traverses an indicated motion path in a time interval that is exactly equal to the communicated activity instance duration. This flexible technique allows for two vital things. 1) Any arbitrarily shaped velocity profile resulting from any DES model-defined or externally performed computation can be explicitly applied to a moving simulation object, and 2) the temporo-spatial control of all simulation objects remains entirely with the underlying simulation model since the animation methods merely scale a defined velocity profile to fit the duration of a communicated activity instance.

For any defined motion path trajectory, a simulation model defines the shape of a desired object velocity profile by specifying an arbitrary number of velocity-distance pairs. The specified velocity values can span any positive numerical range and the corresponding indicated distances are the percentile (0 to 100) arc lengths along the path. The shape of the profile is deduced by plotting the path's percentile arc distance on the abscissa and the corresponding velocity values on the ordinate. This is graphically presented in figure 2. No limitations are placed on this definition except that the distance value in the last specified velocity-distance pair must equal 100 percent (i.e. the current path's total arc length).

As a first pre-processing step, the defined velocity versus percentile distance profile is converted to a velocity versus actual distance curve. As figure 3(a) depicts, this is accomplished by simply replacing the percentile arc lengths on the abscissa by the corresponding actual arc distances for the current path. The specified velocity values are left unchanged. The total time T_o, required to traverse

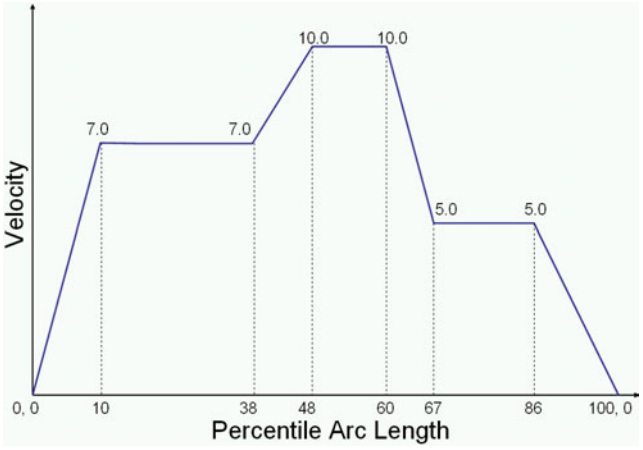


Figure 2: Definition of Velocity Profile Shape

this converted profile in its unmodified form can then be given by the following equation:

$$T_o = \frac{S_1}{V_{o_1}} + \frac{S_2}{V_{o_2}} + \frac{S_3}{V_{o_3}} + \ldots \frac{S_n}{V_{o_n}} = T_{o_1} + T_{o_2} + T_{o_3} + \ldots T_{o_n} \quad (1)$$

As figure 3(a) indicates, $V_{o_1}, V_{o_2}, V_{o_3}, \ldots V_{o_n}$ are the average velocities at which the respective path segments $S_1, S_2, S_3, \ldots S_n$ are traversed. In the case of this converted, unmodified curve, the total original travel time works out to be 195.66 seconds. Now, for any communicated activity instance of duration T_i (say 150 seconds), the described velocity profile is segmentally scaled up or down such that a simulation object traveling that path with the resulting modified (i.e. scaled) velocities reaches the end of the path in exactly T_i time units. This is graphically depicted in figure 3(b). This scaling procedure can be described as:

$$\frac{(T_{i_1} + T_{i_2} + T_{i_3} + \ldots T_{i_n})}{T_i} = \frac{(T_{o_1} + T_{o_2} + T_{o_3} + \ldots T_{o_n})}{T_o}$$

$$\therefore (T_{i_1} + T_{i_2} + T_{i_3} + \ldots T_{i_n}) = \frac{T_i}{T_o}(T_{o_1} + T_{o_2} + T_{o_3} + \ldots T_{o_n})$$

$$= T_{o_1}\left(\frac{T_i}{T_o}\right) + T_{o_2}\left(\frac{T_i}{T_o}\right) + T_{o_3}\left(\frac{T_i}{T_o}\right) + \ldots T_{o_n}\left(\frac{T_i}{T_o}\right) \quad (2)$$

The underlying assumption that is made in segmentally scaling the original converted velocity profile (figure 3(a)) to accommodate the currently specified activity instance duration (figure 3(b)) is that:

$$T_{i_1} = T_{o_1}\left(\frac{T_i}{T_o}\right), \quad T_{i_2} = T_{o_2}\left(\frac{T_i}{T_o}\right), \ldots \quad T_{i_n} = T_{o_n}\left(\frac{T_i}{T_o}\right) \quad (3)$$

With this assumption, the equation that distributes the communicated activity instance duration over the different velocity segments can be written as:

$$T_i = T_{i_1} + T_{i_2} + T_{i_3} + \ldots T_{i_n} = \frac{S_1}{V_{i_1}} + \frac{S_2}{V_{i_2}} + \frac{S_3}{V_{i_3}} + \ldots \frac{S_n}{V_{i_n}} \quad (4)$$

The fact that:

$$T_{i_1} = \frac{S_1}{V_{i_1}}, \quad T_{i_2} = \frac{S_1}{V_{i_2}}, \ldots \quad T_{i_n} = \frac{S_1}{V_{i_n}} \quad (5)$$

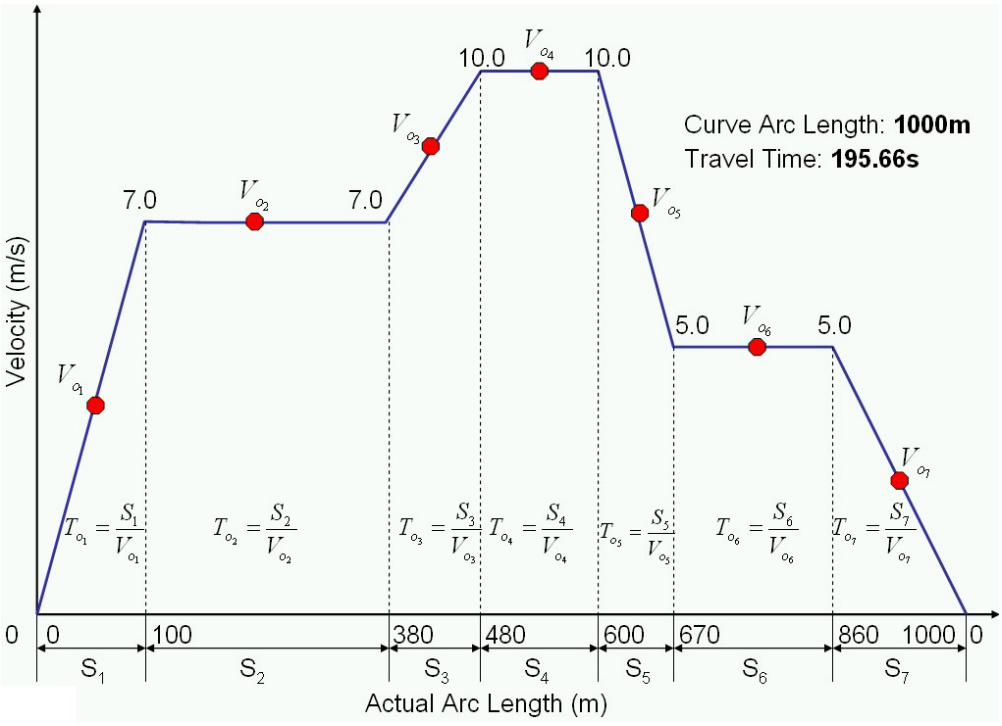
provides the basis for computing the average segmental velocities $V_{i_1}, V_{i_2}, V_{i_3}, \ldots V_{i_n}$, from which the scaled, modified, activity instance-specific velocity profile can be constructed (figure 3(b)). A simulation object that follows this modified velocity profile is guaranteed to traverse the path in the exact communicated activity instance duration T_i.

The following section presents the implementation of the described computation scheme. The implementation is a powerful tool that allows engineers to accurately describe the 3D motion of virtual DES objects on realistic-looking, smoothly-curved motion trajectories.

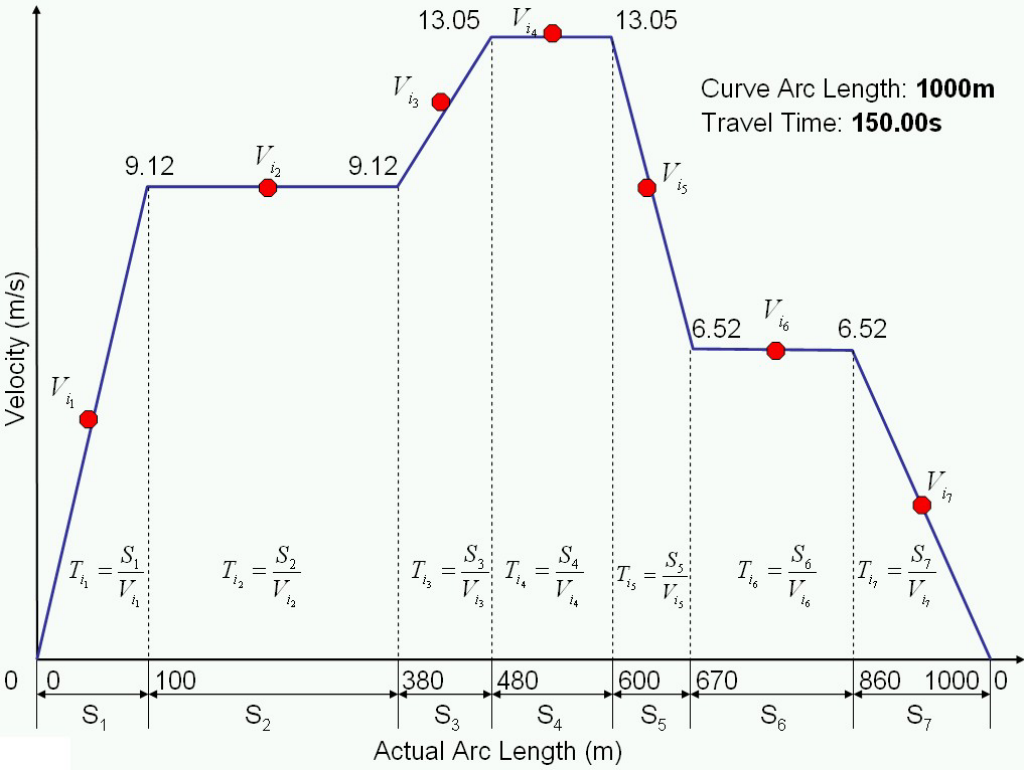
3 PATHFINDER

The algorithms that allow engineers to define flexible, smooth, curved motion path trajectories and then move virtual simulation objects on those trajectories with desired velocity profiles are implemented as a powerful software tool named PathFinder. This tool has been designed as an extension to the VITASCOPE visualization system. VITASCOPE is a user-extensible 3D animation language designed specifically for visualizing simulated processes (particularly construction operations) in smooth, continuous, dynamic 3D virtual worlds.

In particular, the PathFinder add-on defines parametric text animation language statements that allow the definition and manipulation of smooth, curved motion trajectories of arbitrarily complex shapes. PathFinder also implements statements that allow simulation models to 1) specify a default per-path arbitrary velocity profile shape, and 2) over



(a) Original Velocity Profile



(b) Adjusted Velocity Profile

Figure 3: Derivation of Time-Scaled Velocity Profiles

ride (if necessary) the de fault velocity profile shape for any communicated instance of simulation object motion.

In addition, PathFinder also implements terrain-following techniques that help orient moving objects correctly in 3D, and the velocity profile scaling methods that heuristically resize the current velocity profile shape to accommodate the duration of a communicated activity instance. PathFinder thus presents the technologies that allow engineers to accurately describe the 3D motion of virtual discrete-event simulation objects on realistic-looking, smoothly-curved motion trajectories.

Figure 4 presents an animation trace with statements that define a smooth motion trajectory and then move an object over it with an indicated velocity profile. We define motion trajectories in 2D planar resolutions by specifying a series of 2D (i.e. *x* and *z* coordinate only) control points and then manipulate (if needed) the shape of the resultant spline curve by adjusting the tension, continuity, and/or bias of one or more knots (i.e. control points). A default velocity profile (if any) can also be specified as part of a path's definition. In the absence of an explicitly indicated velocity curve for a path, PathFinder assumes a default, constant-speed profile for objects that traverse that path. The defined 2D trajectory is then superimposed on the 3D terrain model that represents the underlying simulated system's landscape.

Simulation objects can obviously override a path's default profile during an activity instance (i.e. motion) communication. This can, for instance, allow simulation models to specify a unique velocity profile that is a function of the properties (e.g. engine power, loaded mass) of the in-context simulation object (e.g. dumptruck) in a communicated instance of an activity (e.g. haul dirt). When a simulation object is instructed to move on a particular path in the specified simulation time units, the current (path default or object overridden) velocity profile is appropriately scaled such that the time to traverse the path with that profile is equal to the communicated activity instance duration. The simulation object then traverses the superimposed 3D path trajectory while following the terrain surface as closely as possible.

Figure 5 presents a strip of animation snapshots taken during the visualization of the motion statement from the animation trace in figure 4. The 3D projection of the described 2D trajectory passes through highly uneven terrain. However, as the snapshots depict, PathFinder's terrain-following algorithms ensure that the object (i.e. dozer) is correctly oriented on the terrain as it travels the path with the scaled velocity profile. The snapshots presented are not successive computer frames observed during visualization. The discretely captured frames are displayed in a filmstrip format merely to depict a sense of motion. The smooth motion of the dozer during visualization and its non-constant velocity cannot be fully captured in static snapshots. Only the animation can convey that information.

3.1 Describing Object Motion with Terrain-Following

Defined and modified 2D motion path trajectories are converted to their 3D representation by projecting them onto

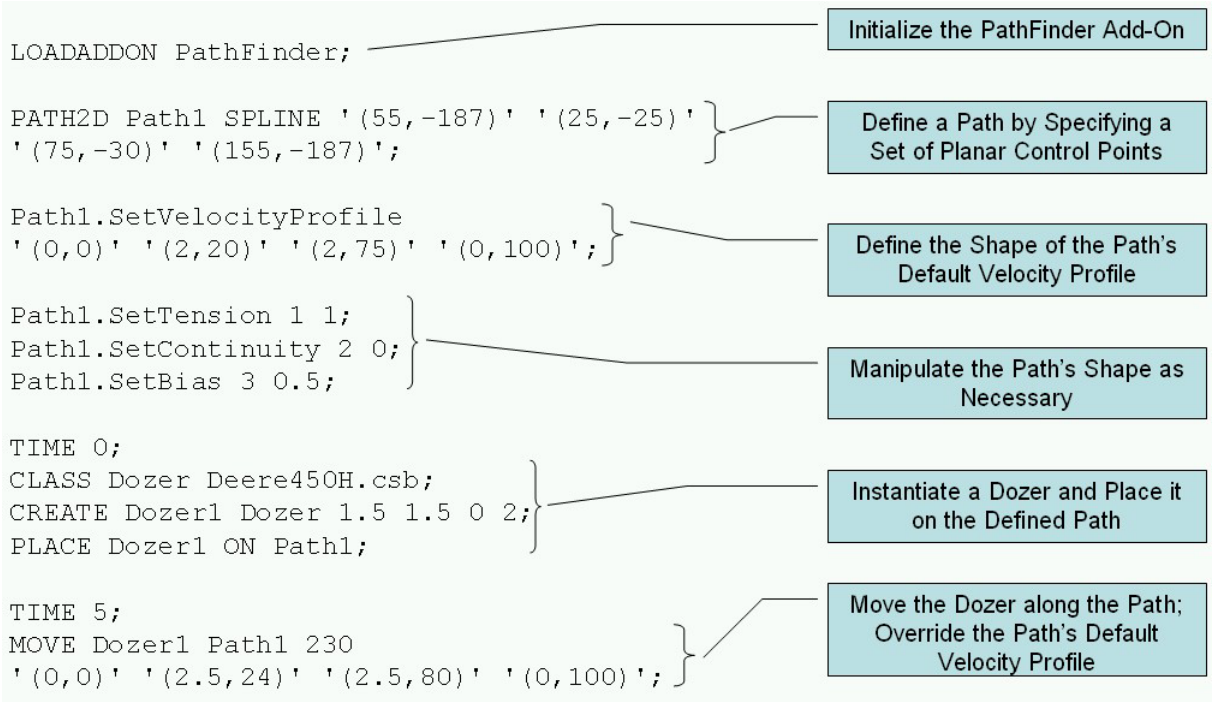


Figure 4: Moving Simulation Objects on Defined Motion Paths

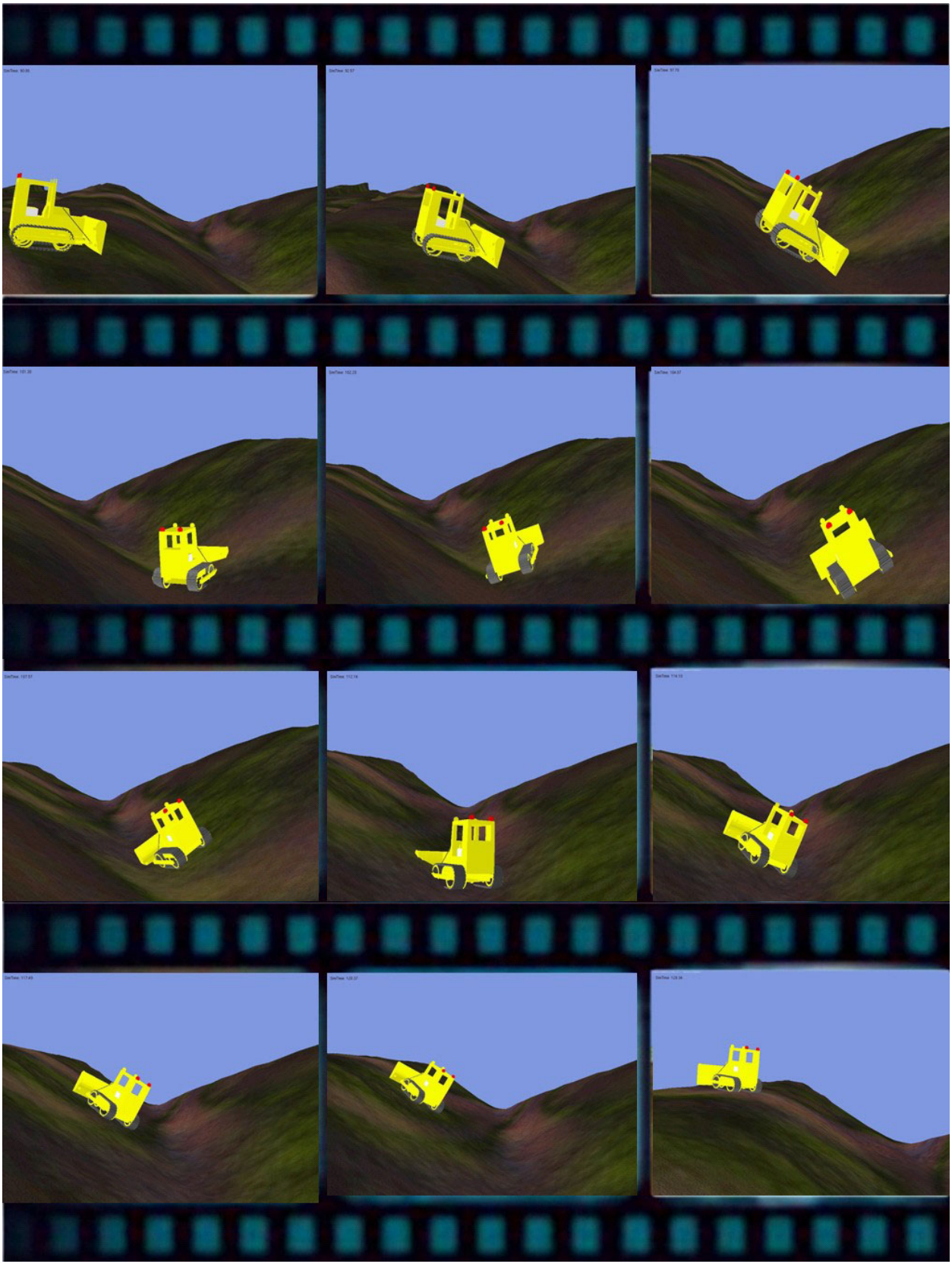


Figure 5: Animation Snapshots of Terrain-Followed Object Motion

the 3D terrain model that describes the simulated system's landscape. At any instant, a moving object's current, adjusted velocity profile dictates the downstream distance on a path at which it is currently located. Given the current downstream distance, the object's planar position (i.e. x and z coordinate) is determined by parametrically retrieving (on the defined 2D path) the point which corresponds to that arc length (i.e. downstream distance). Then, the object's current yaw is calculated by simply computing the tangent (in the 2D plane) to the curved path trajectory at that determined position (i.e. x and z coordinate).

To determine the 3D position of the object inside the virtual world, the computed planar position point is projected on the 3D terrain model of the simulated system's landscape i.e. on the superimposed 3D path. This is accomplished by retrieving the elevation (i.e. height) of the terrain model at that horizontal plane location i.e. we retrieve the y coordinate of the terrain point that corresponds to the x and z values of determined 2D position. This describes the traveling object's current 3D position at that time instant. Since this computation is performed dynamically at visualization run-time, the 2D path's 3D projection always drapes the terrain surface even if its shape deforms during animation i.e. the procedure always retrieves the current terrain elevations (heights) below 2D position points.

Given the 3D position and yaw of the moving object on the terrain surface, the goal of terrain-following is to now orient that object correctly (i.e. compute the pitch and roll) such that its virtual contact points (e.g. a truck's tires) all touch the terrain's surface as closely as possible. To keep moving objects correctly oriented on a terrain's surface, it is necessary to find the locations where that object's contact points touch the virtual terrain. This can be done using geometric collision detection techniques. However, this is inefficient because general collision detection inherently involves more complex, CPU-intensive computations than merely computing an object's terrain contact points (Barrus and Waters 1997).

To enforce terrain-following in a moving simulation object, we adopt a generalized technique that systematically computes that object's pitch and roll by projecting its contact points on the terrain model in a manner similar to that used in computing the 3D position. In particular, the 2D positions corresponding to an object's surface contact points (e.g. tires, crawler edges) are projected onto the 3D terrain model to determine the positions where they intersect the surface.

The pitch of the object is calculated first by determining the mean terrain elevation (i.e. height) along the object's front and rear edges. In particular, the mean elevation along the front edge can now be obtained by simply averaging the y coordinates (i.e. heights) of the calculated front contact points (e.g. front tires). The mean elevation at the object's rear edge is similarly the average of the y coordinates of the rear contact points. The object's pitch at that animated instant is then given by the direction (i.e. vertical orientation) of the 3D vector constructed by joining the computed mean elevation points at the object's front and rear edges.

The side roll of the object is finally calculated using an exactly similar procedure. In particular, the side roll is given by the direction of the vector constructed from the mean elevation positions along the object's left and right edges. These individual computation steps are obviously not visible during an animation. At each animated instant, the object is drawn on the screen in its final, fully-oriented position. PathFinder thus computes a moving object's accurate 3D configuration by prudently synthesizing inputs from the defined 2D motion path trajectory and the terrain model on which the simulated operations are animated.

4 FUTURE WORK

The variable speed motion of simulation objects traversing motion trajectories during the performance of communicated activity instances is purely based on kinetics. In particular, no physical constraints (e.g. mass of an object, its locomotive power, grades of the terrain, gravity) are considered in the computation that describes the simulation object motion. Any depicted 3D process (e.g. a loaded truck continuously accelerating uphill a steep haul road) is a faithful representation of the information communicated by an underlying DES model regardless of whether that process (accelerating when traveling uphill loaded) can be accomplished in real life. Methods that can provide such feedback on physically-impossible simulated processes during visualization can, however, be of significant help within a framework intended to validate modeled (and animated) processes. Future work can explore techniques of designing such methods by incorporating dynamic physical variables in computing the motions of simulation objects.

In addition, the time-based velocity profile shape scaling techniques designed in this work assume the profiles to be composed of piecewise-linear segments. This guides the procedure used in scaling the original defined profiles to accommodate communicated activity instance durations. However, typical velocity profiles generally exhibit curvature, particularly in the acceleration and deceleration phases of a moving object. Future work can explore techniques of defining such curved velocity profiles and design methods for heuristically scaling them to accommodate different motion completion times.

5 CONCLUSION

The presented research extends the state-of-the-art of scientific 3D visualization of DES modeled processes. The work puts in place the techniques that engineers can use to instruct virtual simulation objects to follow any arbitrarily-shaped velocity profiles while adhering to fixed motion

completion times when traversing along any defined motion path trajectories.

Discrete-event simulation models, by their very nature, are unconcerned about the rate at which activities in a model are performed. They only enforce the time instants at which activity instances start and end. Since the temporo-spatial control of virtual simulation objects cannot be wrested away from DES models, any information that defines an object's variable velocity profile must originate within an underlying DES model. A computation scheme that allows DES models to define the general shapes of relevant velocity profiles and then heuristically scales those profiles to accommodate communicated activity instance durations performs well in a framework for animating DES models.

Such a technique not only allows simulation objects to be moved with any arbitrarily shaped velocity profiles, but also ensures that their temporo-spatial control rests entirely with the underlying DES models. The visually accurate animation results we obtain prove that this is not only possible, but also very effective in convincingly presenting modeled operations in dynamic 3D virtual worlds.

ACKNOWLEDGMENTS

The presented work has been supported by the National Science Foundation (NSF) CAREER and ITR programs. Any opinions, findings, conclusions, and recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

Barrus, J. W., and R. C. Waters. 1997. QOTA: A Fast, Multi-Purpose Algorithm for Terrain Following in Virtual Environments. In *Proceedings of the Second Symposium on Virtual Reality Modeling Language,* 59-64, New York, New York: Association for Computing Machinery.

Kamat, V. R., and J. C. Martinez. 2003a. Automated Generation of Dynamic, Operations Level Virtual Construction Scenarios. *Electronic Journal of Information Technology in Construction (ITcon),* Vol. 8, 65-84. Available online via <http://www.itcon.org> [accessed June 20, 2003].

Kamat, V. R., and J. C. Martinez. 2003b. Validating Complex Construction Simulation Models Using 3D Visualization. *Systems Analysis Modelling Simulation,* Vol. 43:4, 455-467.

Law, A. M., and W. D. Kelton. 2000. *Simulation Modeling and Analysis,* 3rd Ed. New York, NY: McGraw-Hill.

Schriber, T. J., and D. T. Brunner. 2001. Inside Discrete-Event Simulation Software: How it Works and Why it Matters. In *Proceedings of the 2001 Winter Simulation Conference,* ed. B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, 158-168, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

AUTHOR BIOGRAPHIES

VINEET R. KAMAT is an Assistant Professor in the Department of Civil and Environmental Engineering at the University of Michigan. He received his PhD in Civil Engineering at Virginia Tech in 2003; an MS in Civil Engineering at Virginia Tech in 2000; and a BE degree in Civil Engineering from Goa University (Goa, India) in 1998. He designed and implemented the VITASCOPE visualization system with J. Martinez as part of his doctoral research. In addition to visualization, his research interests include discrete event simulation, and decision support systems for construction. His email and web addresses are vkamat@umich.edu and <http://www.engin.umich.edu/~vkamat>.

JULIO C. MARTINEZ is an Associate Professor in the Via Department of Civil Engineering at Virginia Tech. He received his PhD in Civil Engineering at the University of Michigan in 1996; an MSE in Construction Engineering and Management from the University of Michigan in 1993; an M.S. in Civil Engineering from the University of Nebraska in 1987; and a Civil Engineer's degree from Universidad Catolica Madre y Maestra (Santiago, Dominican Republic) in 1986. He designed and implemented the STROBOSCOPE simulation language with P. Ioannou and was V. Kamat's research advisor. In addition to discrete event simulation, his research interests include construction process modeling and decision support systems for construction. His email and web addresses are <julio@vt.edu> and <http://strobos.ce.vt.edu>.