Automatic generation of Complex Bridge Construction Animation Sections by coupling Constraint-based Discrete-Event Simulation with Game Engines

Gergő Dori, Dipl.-Ing.
Technische Universität München, Computational Modeling and Simulation Group
dori@bv.tum.de, http://www.cms.bv.tum.de

André Borrmann, Prof. Dr.-Ing.
Technische Universität München, Computational Modeling and Simulation Group
borrmann@bv.tum.de, http://www.cms.bv.tum.de

ABSTRACT: *Construction animations make it possible to conduct detailed analyses of geometric constellations in dynamic construction processes and therefore form an excellent basis for feasibility studies. Construction animations are, however, usually generated manually and typically time-consuming and therefore costly. In this paper we introduce a new method for automating the generation of construction site animations. The method is based on "animation snippets" which represent individual work steps that can be assembled to form a complete animation sequence. The animation snippets are parameterized with respect to geometry and time so that they are adaptable to the specific conditions of a particular construction process. In the proposed concept, these parameters are set by the output of a constraint-based discrete event simulation. The animation snippets themselves are realized using a game engine. The entire methodology is illustrated using the example of a bridge construction process. An animation snippet used to model a crane transporting concrete is discussed in detail.*

KEYWORDS: *construction site animation, game engine, discrete event simulation, preparator, animation snippet, crane animator, ray-tracing*

1. Animation of construction processes

Virtual reality animations of construction sites combine comprehensive 3D models with dynamic information, i.e. they display the movements of individual machines, vehicles or even workers. With the help of such animations, it is possible to conduct detailed analyses of the geometric constellation of the construction site as well as to investigate the construction processes. Feasibility studies conducted before the construction project starts make it possible to identify problems in the planned construction workflow, including potential clashes between machines and/or building elements as well as erroneous sequences of work steps. Thanks to the advanced immersive features of modern VR systems, the user can be observe the construction site from almost any position, providing a better understanding of the construction workflows.

1.1 Process of automation

Today, construction animations are typically produced manually in a laborious and time-consuming process. Their high cost of production means that they are not an option for most construction projects. This is closely related to the fact that construction projects are in most cases unique, as both the resulting buildings as well as the conditions vary from project to project. In this respect, the construction industry differs from other sectors where the number of items produced is much larger and additional effort in the pre-production phase pays off if it is able to improve the production process. To make the generation of animations more economical for the construction industry, a higher degree of automation is necessary.

Our approach for achieving a higher degree of automation consists of three phases (Fig. 1):

- First phase (preparations): A 3D model of the building is enhanced with process information through the application of construction methods. This is achieved using a dedicated preprocessing toolkit called Preparator (Wu et al., 2010, Dori et al. 2010). The output of the Preparator is a large set of individual work steps necessary to erect the respective construction.

- Second phase (simulation): A detailed process simulation is run to find a near optimal sequence of construction processes. As a result, each work step is associated with its calculated start and end date.

- Third phase (animation): An animation of the construction processes is generated using a game engine. To facilitate automation of this phase, *parameterized predefined animation snippets* are created for individual tasks, e.g. move, turn object, etc.

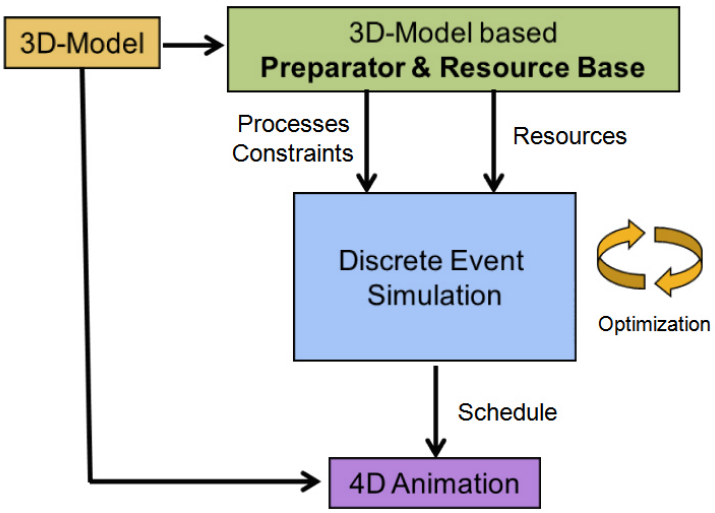

Fig. 1: Schematic overview on the components used for the automated generation of construction site animations

2. Preparations

To visualize an animation of the construction processes, a detailed *3D model* is required that comprises the required construction components as well as the environment of the construction site. To automatically create an animation, three *primary parameters* have to be considered: *what* component should be built *when* and *how (animation type, e.g. movement, rotation)*. The *secondary parameters* are those required to create the defined animation type and can differ from one another in each case, e.g. different parameters are needed for a linear movement than for rotation.

2.1 What, when, how?

The definition of the parameters that concern the *what*, and in part also the *how*, is realized using a toolkit called *Preparator* (Wu et al. 2010, Dori et al. 2010). With the help of this software the user assigns predefined construction methods to individual components of the 3D model, implicitly defining individual *works steps* as well as their resource and precedence constraints. In a later phase (animator), these work steps will be associated with a set of *animation snippets* used to visualize the construction of the respective component. An animation snippet models an individual action that can be well defined (movement, rotation, scale, create new object, change transparency, etc.). A work step is therefore usually composed of a number of animation snippets, as is the case for *fill with concrete*, for example:

1. Move concrete mixer to the construction site → 2. Take funnel hopper from the storage area → 3. Move funnel to the concrete mixer → 4. Fill the funnel with concrete → 5. Move the funnel above the object → 6.a Release the concrete from the funnel, 6.b Fill the object → 7. Move funnel to storage area/delete funnel → 8. Move concrete mixer (leave construction site).

To define the dependencies and requirements of the individual work steps, they are connected by different kinds of constraints. The two basic categories are:

- *Precedence constraints*, which are necessary to define sequential time dependencies between work steps, e.g. formwork and reinforcement has to be assembled before filling the item with concrete.

- *Resource constraints*, which define the required resources to complete the respective work step, such as manpower, machines, materials etc.

As input, the Preparator uses a 3D model of the entire construction stored in VRML format. When the user selects a construction method for an individual component and defines the work steps and their associated constraints, the Preparator isolates the VRML code of the respective component and saves it in a new file (Fig. 2). This is required for later animation, where scenes are composed of individual objects that are moved independently of each other.

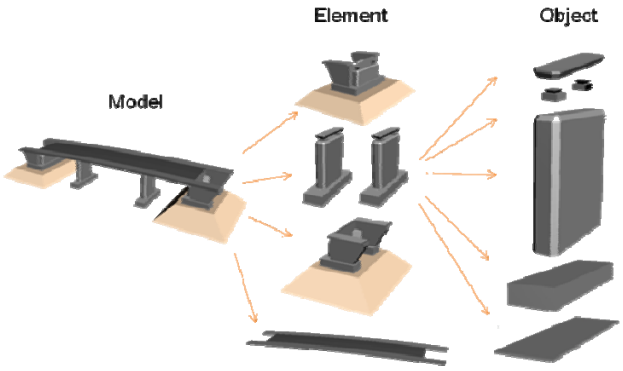

Fig. 2: Decomposition of a 3D model to elements and objects

The "when" parameters for the animation, i.e. when each individual work step starts and finishes, are derived by a simulation of the construction process. To obtain accurate results it is necessary to predefine and import not only the work steps but also the *available resources* into the simulation.

2.2 Resources

The Resource Base is an extension of the Preparator. With this tool the available resources can be defined and exported for use in the simulation. These resources play an important role in the simulation: the required workers, machines and materials are selected and reserved from the stock of resources available for each individual work step during construction, and the duration of a single work step is calculated based on the capacity and performance of the workers and/or machines. While one work step is being undertaken, the resources used are therefore not available for other purposes. The Preparator defines the parameters for the animation of a machine, worker or material. During simulation, an animation snippet is connected with the resource (3D object) for the duration of the period that the simulator determines it is required.

The file exported from the Preparator and the Resource Base serves as the input data for the simulation, and later (extended with the results of the simulation) also for the animation.

3. Constraint-based discrete-event simulation

As described above, one of the most important parameters for the animation snippets are the start and end dates of the individual work steps. For small construction sites, these dates can be calculated fairly accurately with conventional available schedule optimization techniques. But for large-scale construction projects, the creation of an optimized schedule becomes difficult or even impossible. In such cases, modern simulation techniques can be very effective and lead to both time and cost savings.

3.1 Concept of the simulation

Conventional approaches for simulating construction processes employ the discrete-event based method (AbouRizk et al. 2010). The discrete-event simulation technique was originally developed for the mechanical engineering and automotive industries where the product components typically travel along fixed assembly lines and are handled by workers or processed by machines at specific points (workstations). Construction projects, by contrast, do not have fixed activity sequences but are much more dynamic and spontaneous. To incorporate this quality into discrete-event simulation, a *constraint*-based extension has been developed (Beißert et al. 2008a). The constraints define pre-requisites for the execution of a work step. These include the required resources or technological dependencies (see Section 2). If there are several executable work steps possible in one segment of time, we can select one either randomly or according to a certain strategy (e.g. as fast as possible, cost saving etc.). Strategies are implemented by using *soft-constraints* (Beißert et al. 2008b) which make it possible to define

priorities in the order of executable work steps. To find a near optimal solution for the schedule a Monte-Carlo analysis is performed, either with the same, or with different process configurations (resources, strategies).

3.2 Result of the simulation

In the first step, the simulation program reads in the file exported from the Preparator, importing the work steps and resources as well as precedence and resource constraints. The work steps are created in the simulation as *work packages* which are connected to their *attributes* such as required resources, properties of the object etc. The checking process of the resource constraints as well as the calculation of the duration of the work step is performed using these attributes (volume, pieces, area etc.), and the corresponding attributes of the resources (class, performance etc.). When a work step is executed, the start and end dates are determined and saved back into the same file that was imported at the beginning of simulation. As a result, each individual work step is now associated with a part of the 3D geometry in the model, a start and an end date. What remains is how it should be animated.

4. Animation

To create the animation snippets a tool called *Animator* is used. It loads the work steps, reads in the information about the animation, and using the necessary parameters creates the respective animation snippet. This snippet will be stored in a list and as the animation proceeds the animator checks which animation snippet should be started. In this way a complete animation of the construction site can be created.

By way of example, we shall examine the animation of the process of concreting and the associated crane movements. In many construction projects, these processes are critical to avoid any potential spatial collisions and their impact on the overall project duration. A detailed investigation of these processes can help reduce costs and ensure timely completion of the project.

4.1 Game engine

For creating construction animations we use Irrlicht, an open source game engine [Irrlicht]. In general, game engines are development platforms which are highly optimized for visualizing and interacting with 3D scenes. They usually support real-time rendering and integrate a physics engine for real-time collision detection, ray-tracing, gravity field simulation, etc. These features can also be employed for realistically visualizing and animating construction site processes, enabling the user to be placed virtually in the model. The advantages of using game engines for visualizing construction processes have been reported in (Yan et. al. 2011), where Microsoft XNA was used for interactive architectural visualization, and in (Nikolic et. al. 2010), where the same engine was used for the Virtual Construction Simulator, an educational tool for teaching students construction scheduling tasks. ElNimr and Mohamed (2011) used Blender and TrueVision 3D to create a simulation driven visualization of construction operations.

4.2 The animation

4.2.1 Parameters

The *primary parameters* of an animation are those that apply for all the animations: what component is animated, when, and how (Section 2.1). While the Preparator defines the objects and the simulation the time parameters, the “how” parameters are defined by the Animator. The necessary parameters of each animation snippet are the *secondary parameters*. These can be coordinates, rotation angles, state of the object (light, reflection, transparency, diffusion etc.), whether it follows an object and so on, i.e. parameters that are needed to realize the animation. They are stored together with the work steps in an XML file exported from the Preparator, and are extended by the results of the simulation. The Animator reads in this file and the included work steps. It then loads the connected 3D object of the work step into the scene, and creates the appropriate animation snippets using the predefined parameters. When the process finishes, the animation can be started. Using this concept the manual effort of producing an animation is significantly reduced by dramatically increasing the degree of automation. The user needs only define the connection between the 3D object and the construction process, define some extra parameters for movements or rotations, and to initiate the simulation and the animation. The remaining steps all happen automatically.

4.2.2 Hierarchy

We define five levels of animation (Fig 3.) according to the hierarchy of the construction processes generated using the Preparator (Level of detail approach – Wu et al. 2010, Dori et al. 2010). The basic level of the animation is the *animation snippet* (Level 5). A snippet always changes one secondary parameter of the animated object, so it can mean movement, rotation, scale, following another object etc. A group of these snippets together comprise an *animation set* (Level 4). These sets visualize a single work step.

Animation sets / work steps are assembled to form an *animation package* (Level 3) which visualizes a complete construction method that shows how a sub-element is constructed (e.g. create an object made of reinforced concrete). At Level 2 *animation packages* are combined to form *construction elements*. This level is used to visualize the construction of complete elements, for example an entire pier or an entire abutment. At the top level (Level 1), these elements are then connected to form a *complete construction site animation*.

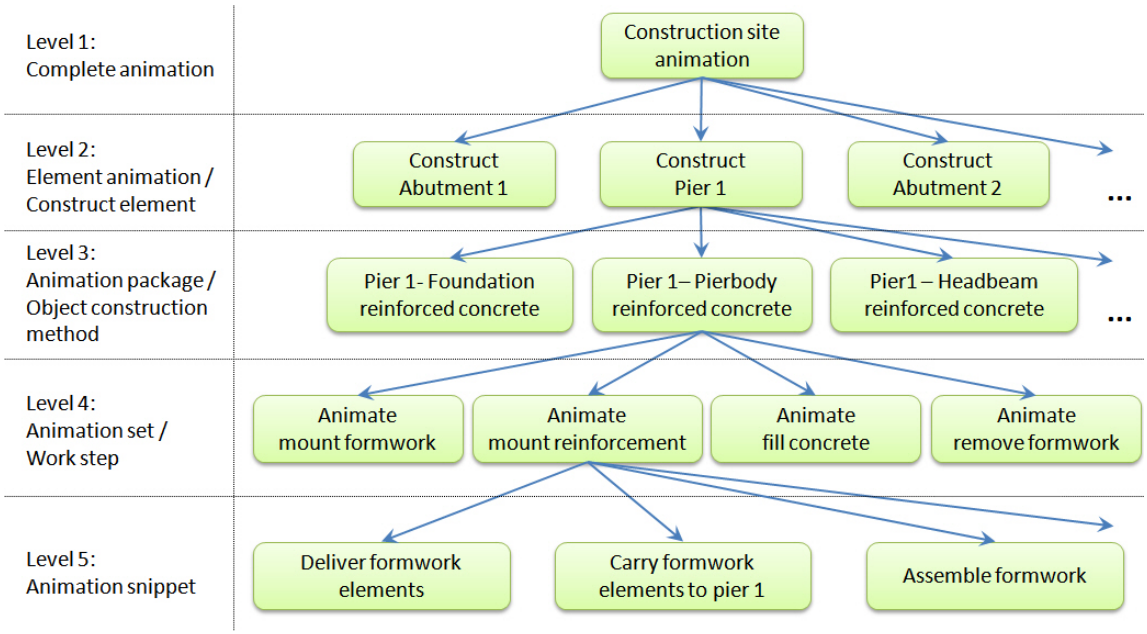

Fig. 3: Hierarchic system of levels of the animation

4.2.3 Animation snippets

An animation snippet is the basic element of the construction site animation and the only element of the hierarchy which is actually animated. The elements on higher levels are formed by assembling combinations or sequences of lower-level steps and do not add additional information to the animation. As an example we show the sequence of animation snippets for the construction method *create an object made of in-situ reinforced concrete*:

1. Delivery of the formwork elements: truck drives to the construction site: normal 3D-movement or Follow-Spline animation (Spline has to be predefined).
2. Transfer the formwork elements one by one to the base object: using Crane animator (section 4.3).
3. Assemble formwork: appearance of the formwork supporter elements / Crane animator.
4. Delivery of reinforcement: truck drives to the construction site: 3D-movement or Follow-Spline.
5. Transfer reinforcement to the base object according to a predefined assembly order: Crane animator.
6. Retrieve a funnel hopper for concreting works from the storage area: Crane animator.
7. Drive concrete mixer to the construction site: 3D-movement or Follow-Spline animation.

8. Transfer the funnel hopper to the mixer: Crane animator.
9. Fill the funnel with concrete: size-changing animation.
10. Transfer funnel hopper above the base object: Crane animator.
11. Filling concrete:
    a. Release concrete from funnel: size-changing animation.
    b. Fill concrete to the base object: size-changing animation.
12. Transfer funnel hopper back to the truck: Crane animator.
13. Loop steps 8-11 until the base object is fully filled with concrete.
14. Return funnel back to storage area: Crane animator.
15. Drive concrete mixer away: 3D-movement or Follow-Spline animation.
16. Curing time: change surface reflection.
17. Dismount formwork: Crane animator – move formwork elements back to storage area.

An animation snippet is used for each of the individual steps. In this case the construction method (*animation package*) comprises 17 animation snippets. Following the hierarchy, the construction method is made up of four work steps (*animation sets*): *mount formwork – mount reinforcement – fill concrete – remove formwork*. The *mount formwork* animation set is comprised of the animation snippets 1 to 3, the *mount reinforcement* animation set of the snippets 4 to 5, the *fill concrete* set of the snippets 6 to 16, and the *remove formwork* of snippet 17. Accordingly, different work steps may comprise different numbers of animation snippets. The animator knows which animation snippet has to be created for each work step, so all these animation snippets will be defined automatically during the file importing process.

The list provides an indication of the kind of tasks an animation snippet can visualize. The 3D movement and the Follow-Spline animation snippet play important roles. These animation snippets are mostly used for animating vehicles, machines and workers. A 3D movement is a straightforward linear movement between the start and the target point. The Follow-Spline animation is similar, but instead of moving linearly the object follows a spatial curve. It is used, for example, when the surface of the terrain is not flat, or a road is not linear.

The next generation of this animation snippet is the *surface following movement* snippet. For this, gravity field and collision detection methods must be implemented in the model. This kind of animation is familiar from first person shooter games, where the camera (viewpoint) follows the terrain at a predefined height as the user moves (the first-person). This capability is used to ensure that trucks and other machines will automatically stay on the surface, or drive between points on the surface of the terrain.

Another very important animation snippet component is the crane animator. The crane plays a significant role on the construction site and a special tool is created to automate and control the animation of the crane movements.

4.3 Crane Animator

The crane animator facilities the automatic creation of an animation transporting an object from its place of origin to a target coordinate by means of a tower crane. The carrying process is executed along a predefined path, but with real-time collision-detection and path-correction during the ongoing animation. Collision detection and path correction is important when there are other objects moving on the construction site (trucks, machines), or where further cranes are also operating within the same workspace. The input parameters of the tool are the start and target coordinates of the object as well as the technical properties of the tower crane: height, length of the jib, turning speed of the jib, speed of the trolley, lifting speed. Our tool is dedicated to visualize and find a possible spatial path for the crane movement, not considering physical parameters like forces in the cable etc. as discussed in (Chi et. al. 2010).

4.3.1 The structure of the tower crane

To be able to automate the movements of a tower crane various aspects have to be taken into consideration. The

structure of the crane model has to be defined hierarchically to be able to follow every possible movement of the crane and its components. For example when the jib turns the trolley and the hook have to move with it automatically, but not vice versa. This is realized by a hierarchic *master-slave mechanism*, according to the forward kinematics, where the model has an ordered chain structure (Chi et. al. 2007). If a chain element is moved, the sub ordered chain elements will move with it, but not vice versa. The master of the model is the *mast*. All the other parts are defined as the slaves of the mast. A slave object means, if the master moves, the slave moves automatically with it, but if the slave moves, the master stays unchanged. The first slave is the *jib*. The turning point of the jib is located at the section point of the centre lines of the jib and the mast. The next part is the *trolley*, it moves forward or backward along the jib. Setting the trolley as the slave of the jib only results in a rotation when the jib turns, so the relationship has to be extended to also recalculate the new position of the trolley. The *hook* is the slave of the trolley, connected to it with *cables*. The cables are special in that they are slaves of both the trolley and the hook: the start point of the cable is the slave of the trolley; the end point is the slave of the hook. So if the hook moves up or down, the length of the cable changes. When the trolley moves, then the cable must also move along with the hook, and when the hook swings out, the cable swings as well.

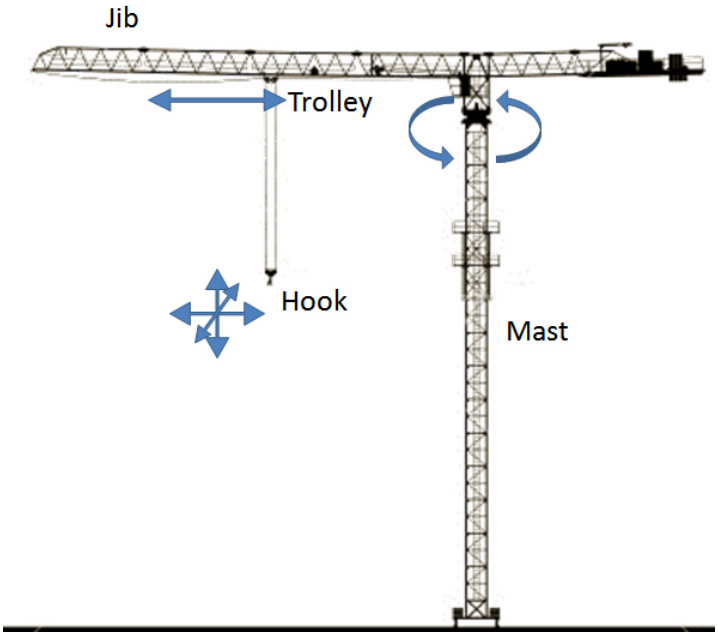

Fig. 4: Basic elements and movements of a tower crane (picture source: iStockphoto)

4.3.2 Carrying process

First the tool checks if both the start and target coordinates are within the range of the crane. If so, the carrying process can be started. The jib turns above the object, lowers the hook and connects the object to it. At this moment the object becomes the slave of the hook. When the hook moves, the object now follows it in every direction. The tool checks the height coordinates of the start and the target points, and lifts the object above the level of the higher one. Then it tries to predefine a possible way between the start and the target point. It checks which is nearer to the crane, and moves the trolley to the nearer position. The trolley always tries to be as near to the mast as possible, so if the start point is nearer, the trolley will stay where it is; if the target point is nearer, the trolley will move to the smaller distance from the crane (Fig. 5 left). The predefined way will be checked for collisions (ray-tracing).

There are two existing general methods for collision detection. The first group can determine whether a collision exists or not, the second one can also return the distance to the nearest possible collision. The second group requires more computation time, however “knowing the shortest possible collision distance can eliminate unnecessary collision checks within the collision free region, significantly reducing the computational time…” (Lai et. al. 2009). To reduce calculation time even more, different strategies are used like usage of different bounding objects (Lai et. al. 2009) or hierarchical ordering of the objects (Chi et. al. 2010). Our ray-tracing approach is somewhere between these two groups. We are attaching different rays to a moving object and check if this ray (with predefined length) is colliding with another object or not, without checking the distance. Doing so, a

collision can be predicted, and the path of the object in time before the collision modified. In case of the predefined path, the path will be converted to a ray and checked if it collides with anything or not. If it is collision-free the movement of the object can be started. If there is an obstacle in the way, a new path will be determined. The area between the object and the target will be discretized with circular lines in the height of the movement looking for collisions (Fig. 5 right). The simulator chooses the nearest collision-free circle from the predefined path and sets this as the new path (when it does not find any free circles, it goes to a higher level and check the circles again). The trolley has to move to this point first along with the object; only then can the jib start to turn in the direction of the target. When it arrives, the trolley moves above the target point, lowers the hook, and places the object at the target coordinates. Consequently the basic predefined horizontal movement is always composed of one linear and one circular component, when it is corrected from two lines and a circular part.

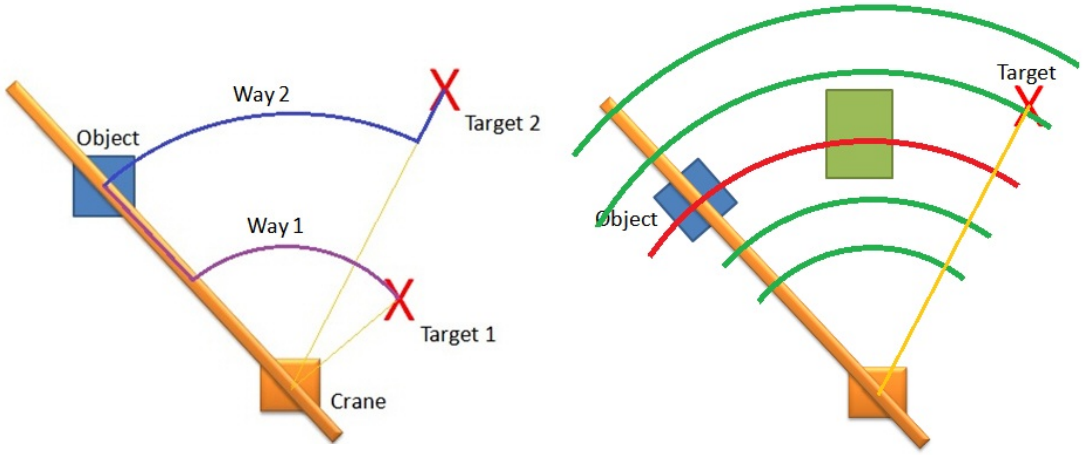

Fig. 5: Carrying ways depending on the distances between the start and target points from the crane (left), and identification of possible collision-free paths using ray tracing

While transporting the object, a second real-time collision detection and route correction routine is performed in order to find and evade other obstacles on the path. These obstacles are mostly moving objects such as vehicles or objects placed by other cranes that were not present when the path was defined, or even the other cranes. For detecting obstacles a ray-tracing technique is used. As soon as a potential collision is detected the crane animator recalculates the path of the object until it is collision-free again, and follows this new path.

The same technique is used for the cranes to detect other cranes in their path. If one receives a message about a possible future collision, the crane stops and tries to communicate with the crane animator of the other crane (which also registered the possible collision). The one with higher priority continues its movement, the other one has to wait, or move back to allow sufficient space for the other to pass. When both cranes have equal priority, one is chosen at random and the other has to wait.

An important factor of the animation is the time interval of a frame. When the crane movement is visualized, the object moves along the predefined path with collision detection and possible path correction undertaken on a frame-by-frame basis. The more frames (= the shorter the time interval) there are to visualize the crane movement, the greater the number of “discrete” steps and the better the chance of detecting a collision. If the time interval between two frames is too great, there is the chance that the animator moves unnoticed through an obstacle between frames. This must obviously be avoided at all costs. Further investigations are necessary to identify the minimum number of frames for crane movement.

5. Summary

In this paper we have introduced a novel approach for automating the process of generating construction site animations. Using the Preparator tool, the basic input data for constraint-based discrete-event simulation and subsequent animation is created by assigning predefined construction methods to individual 3D objects. The simulation is used to determine a possible time schedule, including start and end dates for each individual work step. The animator uses this information in turn to start and stop so-called animation snippets. These represent individual work steps and are assembled to form a complete animation sequence. Animation snippets are parameterized with respect to geometry and time so that they can be easily adapted to respective conditions of a specific construction process.

For the visualization, the game engine Irrlicht is used. Game engines are highly optimized for visualizing and interacting with 3D scenes and usually support real-time rendering and as well as the simulation of basic physical phenomena.

Since cranes play an extraordinarily important role in construction processes, a crane animator tool has been developed that automates the transport path calculation and its subsequent visualization. Using ray-tracing technology in each visualized frame it performs collision detection and if necessary corrects the predefined route. The next step of automation will be to couple simulation and animation. This will allow the user to steer the simulation using the animation, e.g. to pause the simulation, choose the next executable work step, modify the position or operation of a machine or worker, and the simulation will then react accordingly to the changes in the visualization.

6. REFERENCES

AbouRizk S. (2010): Role of Simulation in Construction Engineering and Management, Journal of Construction Engineering and Management ASCE, 1140-1153.

Beißert U.; König M.; Bargstädt H.-J. (2008a): Generation and Local Improvement of Execution Schedules Using Constraint-Based Simulation; XIIth International Conference on Computing in Civil and Building Engineering, Beijing.

Beißert, U., König, M., Bargstädt, H.-J. (2008b): Execution Strategy Investigation Using Soft Constraint-Based Simulation. Proc. of the IABSE Conference on Information and Communication Technology, Helsinki, Finland

Chi H. L., Hung W. H., Kang S. C. (2007): A Physics Based Simulation for Crane Manipulation and Cooperation. ASCE Workshop on- Computing in Civil Engineering

Chi H.L., Kang S. C. (2010): A Physics-based Simulation approach for Cooperative Erection Activities. Automation in Construction 19, 750-761.

Dori G, Borrmann A. (2010): Bauablaufsimulation und -animation für die Planung von Brückenbauvorhaben, 22. Forum Bauinformatik,Berlin, 49-57.

ElNimA. A. r, Mohamed Y. (2011): Application of Gaming Engines in Simulation Driven Visualization of Construction Operations. Journal of Information Technology in Construction (ITcon) Vol. 16., 23-38.

Irrlicht Engine. http://irrlicht.sourceforge.net/

Lai K.-C., Kang S.-C. (2009): Collision Detection Strategies for Virtual Construction Simulation. Automation in Construction 18, 724-736.

Nikolic D., Messner J. I., Anumba C. (2010): The Virtual Construction Simulator: Evaluating an Educational Simulation Application for Teaching Construction Management Concepts. CIB W78: 27th International Conference – Cairo, Egypt

Wu I-C., Borrmann A., Rank E., Beißert U., König M., (2009): A Pattern-Based Approach for Facilitating Schedule Generation and Cost Analysis in Bridge Construction Projects. In: Proc. of the 26th CIB-W78 Conference on Managing IT in Construction. Istanbul, Turkey.

Wu I-C., Borrmann A., Beißert U., König M., Rank E. (2010): Bridge Construction Schedule Generation with Pattern-based Construction Models and Constraint-based Simulation. Advanced Engineering Informatics 24 (4), 379-388.

Yan W., Culp C., Graf R. (2011): Integrating BIM and Gaming for Real-time Interactive Architectural Visualization. Automation in Construction 20, 446-458.