11

Enabling and Analyzing Natural Interaction with Functional Virtual Prototypes

Frank Gommlich, Guido Heumer, Bernhard Jung,
Matthias Lenk and Arnd Vitzthum
Technical University Bergakademie Freiberg
Germany

1. Introduction

Functional validations of virtual prototypes are a promising application area of immersive Virtual Reality (Moehring & Froehlich, 2011). Through the interactive simulation of operating procedures such analyses aim at providing insight about virtual prototypes, e. g. concerning visibility aspects, reachability of control elements, and ergonomics. A key requirement is the enabling of *natural interaction* defined by Zachmann as *"interaction which imitates that same interaction in the real world as close as possible"* (Zachmann & Rettig, 2001). In immersive VR, natural interaction can be realized through data gloves and motion tracking devices.
However, a challenging task remains in the modeling and simulation of interactions with dynamic control elements such as sliders, switches etc. (Moehring & Fröhlich, 2010). We have devised and implemented a framework that not only aims at simplifying the specification of such *control actuators* for interactive virtual environments but also facilitates the recording, automated classification, and analysis of natural interactions with such control actuators. The presented approach builds on the European Standard EN 894-3 (DIN894-3, 2006) and covers all types of control actuators defined therein. This standard defines systematic guidelines for the choice and configuration of control actuators to ergonomically design machinery.
A characterizing feature of control actuators in the real world is given by their function: Controlling the behavior of machines and other appliances. In order to model the triggering of changes of environment objects in the simulation, our framework allows to associate control actuators with *interaction events*. Interaction events may be fired continuously, e. g. during movement of sliders, or when discrete states are reached, e. g. switches. Besides making interactive simulations more realistic, interaction events can also be recorded to allow later playback and analysis of the interaction. For this, detailed timing information is stored in the interaction events. For example, interaction events may be used for the animation of virtual humans that imitate the user interactions when operating virtual prototypes.
Contributions of our research include an XML-based modeling language for control actuators that can be seen as both an abstraction layer and an extension to lower-level rigid body physics modeling capabilities provided e. g. by Collada (Khronos Group, 2008) and X3D (International Organization for Standardization, 2008). Further, we produce a reference implementation of control actuators with direct object manipulation, an interaction database for analyzing interaction events including recognizing basic interactions and grasp classification, and a powerful graphical analysis tool. Through this, the process of adding interactive elements to virtual prototypes can be drastically simplified.

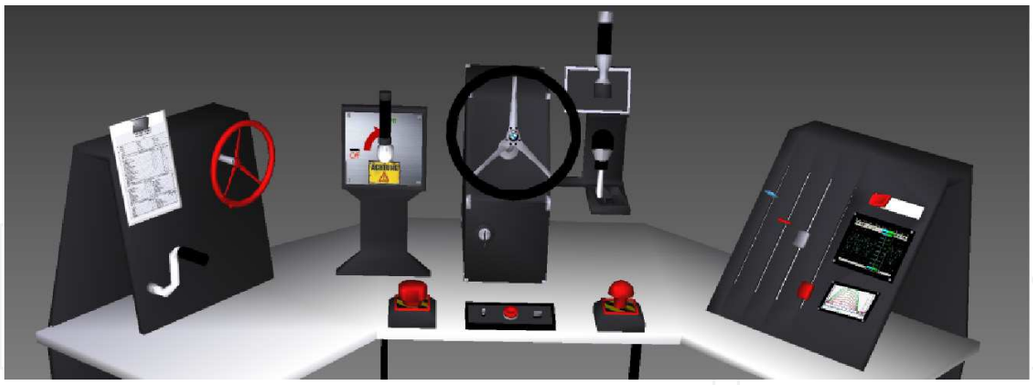


Fig. 1. Selection of control actuators defined in the European standard EN 894-3.

2. Control actuators

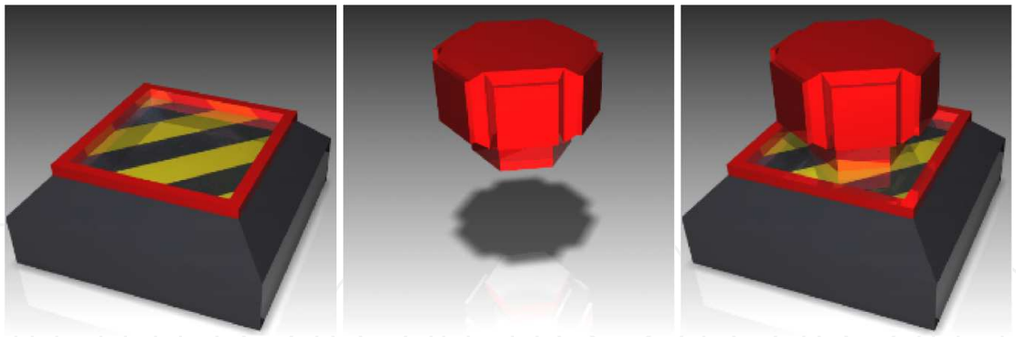
This subchapter introduces the concept of control actuators, our XML based modeling language and a reference implementation of control actuators. An easy way to include different control actuators into a dynamic environment is to use an extended version of annotated objects (Weber et al., 2006). They are declared and managed in an XML-based representation structure, which combines all information about types of scene objects in a common database. Such information includes graphical model, type, component references, physical parameters, joint definitions, etc. Annotated objects are similar to the concept of smart objects (Kallmann & Thalmann, 1999). In contrast to smart objects, they do not contain any form of behavior description, neither for object nor for actor behavior. The object behavior of annotated objects is totally dependent on their physical simulation.

2.1 Concept

Structurally, control actuators are compound objects where a movable part can change its position or orientation w.r.t. a static part, the fitting. The relative movement is constrained by the degrees of freedom (DOFs) of a joint connecting the static and the movable part. Here, the composition of both is called control actuator. In Figure 2 an emergency button is shown. In this example the button takes the role of the actuator. Multiple actuators with the same fitting are also conceivable and are used, e. g., in the case of cockpit instruments, keyboards or other operational controls. Thus, control actuators share many similarities with rigid bodies from the Collada and X3D specifications. However, the definition of control actuators requires additional capabilities not available in basic Collada or X3D. In particular, many types of control actuators such as switches and gear shifts exhibit lock states which correspond to discrete values along a continuous DOF of joints. A control actuator will snap to a discrete state when no further force is exerted on the actuator.
Our concept of control actuators covers all types of control actuators and guidelines defined in the European standard EN 894-3 (see fig. 1).

2.2 Declaration

Annotated objects with their attributes are represented in a compact and easy to use XML structure. To check the correct syntax of annotated objects and their actuators there is an XML schema. To ensure extensibility of the format, all object attributes are denoted by an all-purpose parameter format consisting of name, type and value, represented by the *param* element.



(a) Fitting (b) Actuator (c) Control Actuator

Fig. 2. Composition of a control actuator. Each control actuator consists of a fitting and at least one actuator.

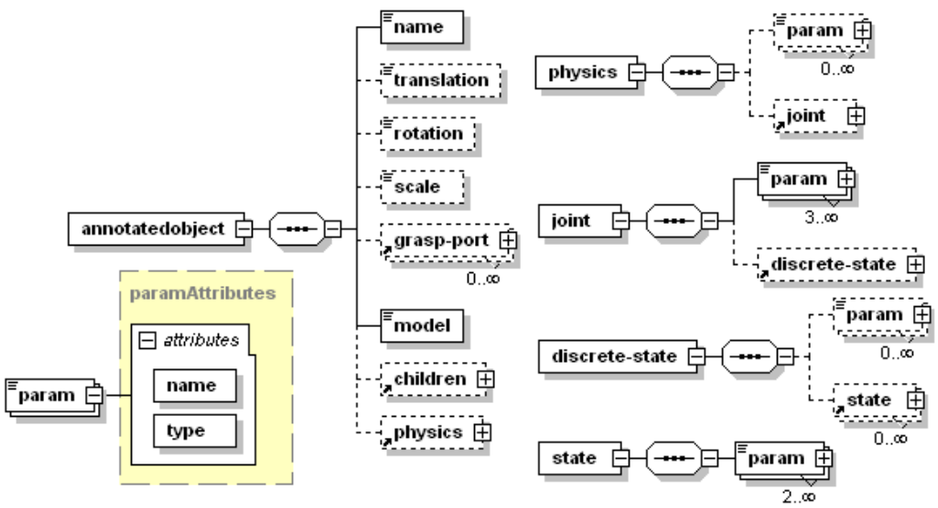


Fig. 3. The XML schema for annotated objects within our framework.

The element *physics* and its subelement *joint* (see figure 3) contains the major characteristics of control actuators regarding their representation in the dynamics simulation. The fitting references one or more actuators with the *children* element. These actuators are also annotated objects with the difference that they have specified information about *joint*s and their constraints. A joint definition contains at least the joint type (e. g. DoubleAxisHinge for a joystick, or Slider for a volume control), the specification of the DOF axes and the constraints along these axes.

In case of discrete actuators it is necessary to define at least one discrete state which is represented by an angular or translational offset from the neutral position. Additional it is possible to specify unsteady intermediate states. When a control actuator reaches an unsteady state, it will automatically move to a defined successor state. This is useful for realizing complex actuator movements, e. g. a gear shift. Figure 4 shows the gears of a car with steady gear states and intermediate discrete states. States Ⓐ and Ⓑ are unsteady with ⑤ as the successor state. Listing 1 shows an example for discrete states of an actuator definition.

It proved possible to implement all presented control actuators in the European standard EN 894-3 with at most two DOFs for either translation or rotation. A combination or mixture of

translation and rotation is not intended by the standard. Therefore it was effectual to divide the standard actuators into four different types according to their DOFs. We called these four joint types *Slider*, *Hinge*, *DoubleAxisSlider* and *DoubleAxisHinge*. The slider and the hinge type have one DOF each whereas the DoubleAxisSlider and the DoubleAxisHinge type have two DOFs.

2.3 Implementation

The implementation of control actuators is based on an open-source physics engine called Newton Game Dynamics (Jerez & Suero, 2011).

In order to effect an actuator manipulation by a VR user, forces are calculated from the user's hand movements and applied to the actuator (see Section 3). The actuator can be continuously moved within its degrees of freedom, constrained by its joint limits and subject to possible collisions with other geometries. When the actuator is released by the VR user, the engine realizes a snap to the closest discrete lock state. The flow of this automatic snap mechanism is shown in Figure 5 and is realized with the help of forces applied to the actuator to reach the calculated position. This results in a smooth and continuous movement to the new position.

Finally, if the reached position is unsteady, then the cycle starts again and the force for the following state will be calculated. This action will be repeated until the first steady state is reached or until the actuator is grasped again.

In order to restrict the actuator movements, e. g. to special tracks in a two dimensional layer (see figure 4, left column) we used the automatic collision response mechanism of the dynamics engine. This mechanism ensures a realistic movement of the stick inside the tracks of the fitting, without explicitly modeling the permissible pathways.

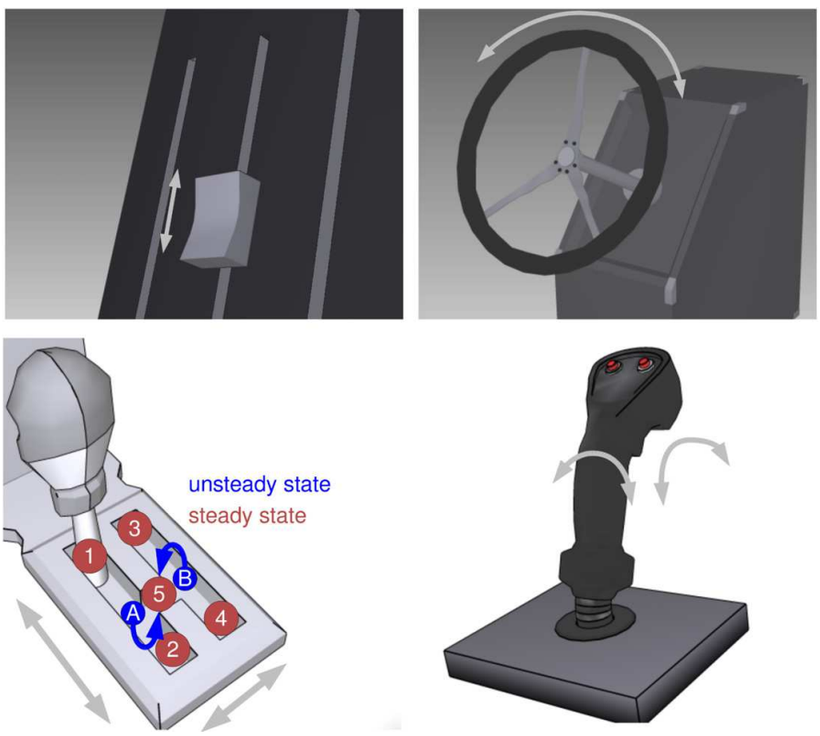


Fig. 4. Overview of the different joint types. first row: Slider and Hinge. second row: DoubleAxisSlider and DoubleAxisHinge.

```
<annotatedobject>
    <name>emergencyFitting</name>
    <model>emergencyFitting.iv</model>
    <children>
        <child type="string">emergencyActuator</child>
    </children>
</annotatedobject>
<annotatedobject>
    <name>emergencyActuator</name>
    <model>emergencyActuator.iv</model>
    <physics>
        <joint>
            <param name="jointtype" type="string">slider</param>
            <param name="pinDir" type="array3">0 0 -1</param>
            <param name="minValue" type="double">0</param>
            <param name="maxValue" type="double">0.1</param>
            <discrete-state>
                <state>
                    <param name="name" type="string">deactivated</param>
                    <param name="value" type="double">0</param>
                </state>
                <state>
                    <param name="name" type="string">activated</param>
                    <param name="value" type="double">0.05</param>
                </state>
            </discrete-state>
        </joint>
    </physics>
</annotatedobject>
```

Listing 1. XML code example of an emergency button, shown in figure 2.

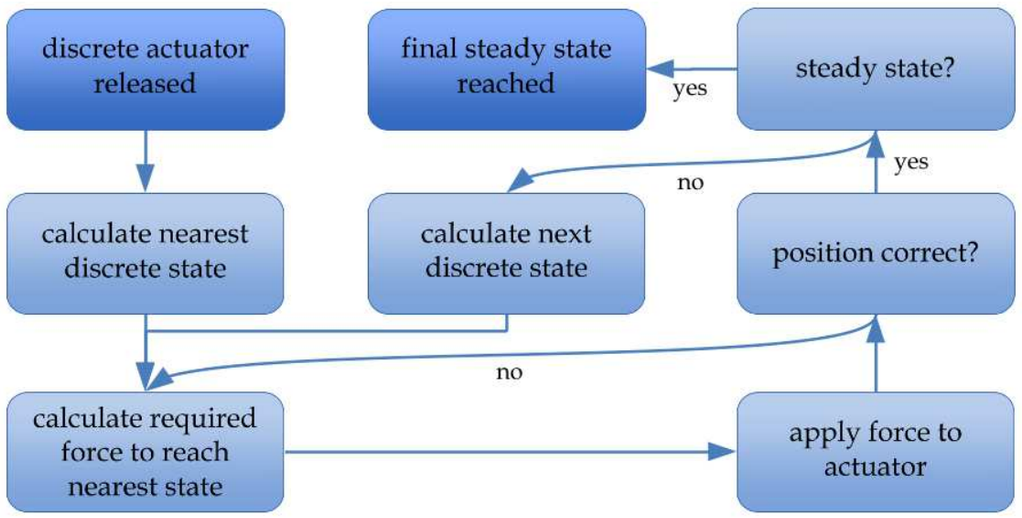


Fig. 5. Flowchart for the snap in behavior of control actuators. If the user releases the actuator, the system moves them to the nearest discrete state.

3. Direct object manipulation

The simulation of realistic manual object manipulation by a user in virtual environments is an important feature in virtual prototyping, ergonomics analysis and virtual training scenarios. To differentiate this kind of natural interaction from other, more commonly used types of object manipulation where the user either selects, grabs and moves an object by help of a pointing geometry (akin to a 3D mouse pointer) or picking ray we refer to it as "direct object manipulation". The discriminating factor is that a user directly controls virtual models of his hands that (more or less) exactly follow his real arm and finger movements (isomorphic control). With the help of these hand models, which act as the representations of the user's real hands in the virtual world, the user is able to touch, push, pick up and drop objects as he would in reality.

Clearly, the functional hand model is an important feature in this kind of interaction technique. The properties of contacts – their location, direction and forces – that exist between a hand and an object determines which kind of manipulation is taking place. E. g. mere touching vs. forceful pushing, prehensile (grabbing) vs. non-prehensile (pushing, dragging, sliding) manipulation, the various types of grasping an object, etc.

To be able to discriminate which parts of the hand are in contact with the object, we work with a setting that uses virtual sensors fitted on the finger segments of the hand model (see figure 6 left). These sensors can detect collision with annotated objects, including the control actuators described in the previous section. Each phalanx (finger segment) is represented by four box-shaped sensors (one on the top, the bottom, the left and the right side) allowing for discrimination of which part of the phalanx collides with the object. Similarly, the palm is represented by differently shaped collision primitives. This approach in constructing the hand model has two advantages. First, it allows for a rather close approximation of collision geometry to the real anatomical shape of the hand while still enabling efficient collision detection by only using convex collision primitives which are supported by many modern implementations of dynamics engines. Second, it provides information about possible contact configurations and whether they form a valid grasp or not, e. g. it is not possible to form a grasp with the inside of the palm and the upside of the thumb. Further, knowing which segment of the hand touches the object helps inform the classification of grasp types (see section 4.2). The sensors themselves are mere collision detectors and have no simulated physical properties. To effect real manipulations on the object, resulting forces are calculated, based on collision centers, collision normals and penetration depths of the sensors and applied to the physical simulation of the annotated objects (see figure 6 right).

Since object behavior is simulated by a rigid body dynamics engine, this approach to implement direct object manipulation simulation enables the user to pick up and release objects and to manipulate them in prehensile and non-prehensile ways.

This allows for *natural interaction* with functional virtual prototypes. Our approach also supports the integration of haptic devices into the hardware setup for an even more realistic interaction experience (Abate et al., 2009; Zorriassatine et al., 2003).

4. Interaction analysis

To be useful in interactive VR applications, a mere simulation of the behavior of control actuators and their interaction with a virtual hand model is in many cases not enough. Additionally, information about interactions of the user with virtual objects and the resulting state changes need to be processed. An important first step is to record all interaction data that is generated during direct object manipulation.

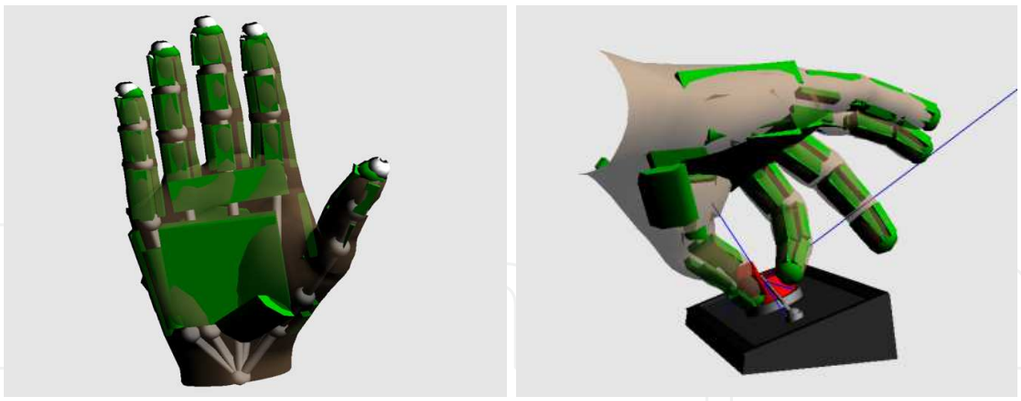


Fig. 6. left: Articulated virtual hand model fitted with collision sensors (depicted in green). right: Sensors of hand model detecting collisions with object and generating respective forces for physical simulation (blue lines).

4.1 Interaction recording

Most importantly, *arm trajectory*, *hand posture* and *collision* data during grasping is recorded. In our system all interaction data is stored persistently in an interaction database organized by recording sessions. Recording sessions are subdivided into channels each of which represents a certain single stream or constituent of the interaction data (see figure 7). This could be an input device like an optical tracker or a data glove, or motion data of a certain part of the body like hand postures or hand trajectories.

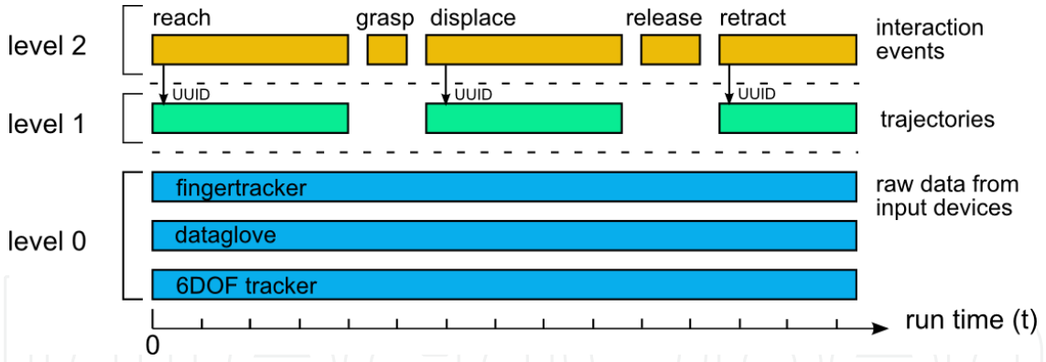


Fig. 7. Examples for channels of the interaction database and their division into to different levels.

Recording channels are organized into different levels, based on the type of data they contain:

Level 0 - Raw data: This level contains the raw data, captured from the different input devices at certain sampling times denoted by timestamps. This data is *continuous* and *homogeneous*, i.e. each input device sends the same amount of data at a time step and continues producing data from the start of the application till the end.

Level 1 - Motion data: On this level, each channel contains motion data of a specific part of the user's body, e. g. hand trajectory, elbow swivel angle, gaze direction, etc. This data also contains time stamps but is additionally segmented into intervals of motion, separated by pauses or by object contact. Each interval can be referenced by higher levels of data via

a unique identifier (UUID), thus enabling playback of interaction events with high visual fidelity while the motion intervals as a whole can be rearranged in time arbitrarily. Thus, in terms of the structure of this data, level 1 channels are described as *homogeneous* and *interval-based*.

Level 2 - Interaction data: Data on this level is the result of detection and classification processes. These are primarily interaction and control actuator events (see section 4.3) which are stored in XML format. The amount and length of data contained in these events is quite variable and they always refer to certain points in time. Thus, data on this level is described as *selective* and *nonhomogeneous*.

Through this multi-level database architecture, persistence of the whole interaction session, from the fine-grained raw data to the abstracted results of the analysis process, is achieved. Further, the data captured in the interaction database can be played back in a modular way. It is possible to play back only a selection of channels, depending on the purpose of the playback. This is where the subdivision into channels and levels becomes valuable. If only level 0 rawdata is played back, the interaction can be simulated again and also the analyses process can be repeated. This allows for training and testing of e. g. the involved classification algorithms. Conversely, if only data from higher levels is played back, animations can be generated reproducing the recognized interactions *in effect*, that is, they can be flexibly adapted to changed scenes or different body model proportions.

4.2 Grasp classification

Hand posture data during hand-object contact is automatically classified w.r.t. its grasp type (Heumer et al., 2008).

During the course of the virtual workers project several grasp taxonomies from the medical, e. g. (Schlesinger, 1919), and robotics literature, e. g. (Cutkosky, 1989), have been explored. Since some significant shortcomings in relation to applicability in virtual environments have been identified, additionally a new taxonomy has been proposed in (Heumer, 2010) which as a unique feature provides support for different types of non-prehensile grasps. The latter play an important role in the operation of control actuators in application domains like virtual prototyping, ergonomics evaluations and simulation of machine operation procedures.

4.3 Recognition of basic interactions

Fed by a multilayer classification and recognition scheme, thoroughly described in (Heumer, 2010), the continuous stream of body movements and scene interactions is segmented and analyzed. As a result, *interaction events* are generated which contain information about smallest semantically meaningful constituents called *basic interactions*, e. g. grasp, touch, push, reach, etc. All basic interactions are characterized by one specific aspect that is modified by the respective type of interaction, such as hand-object distance, hand-object contact, forces, prehension, and object position or orientation. Besides its distinct type or category, a basic interaction is further qualified by a type-specific set of parameters. Figure 8 shows the different types of basic interactions that are currently distinguished. Regarding implementation, the interaction events are realized as observer pattern, so arbitrary other parts of the system can subscribe as listeners to these events such as storage, visualization, action recognition, etc.

A subclass of interaction events are control actuator events (CAEs). These are generated under certain conditions when a control actuator changes its state. The exact condition of generation and the granularity of information contained in the events is configurable. Generally, CAEs are fired when the user interaction with a control actuator ends (see figure 9 for an example).

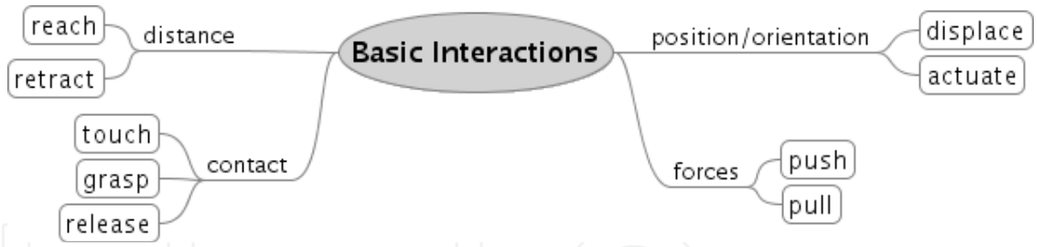


Fig. 8. Taxonomy of basic interactions.

In this case, the event contains the state (rotation around rotational DOFs, translation along translational DOFs) quantified between a minimum and a maximum value. For control actuators with discrete states, the name of the final state is also included. Furthermore, intermediate states during the interaction can also generate events when they are passed before the interaction ends. All state names and DOF information is referenced as specified in the XML actuator declaration (cf. 2.2) thus rendering the event format also human-readable. See listing 2 for an example.

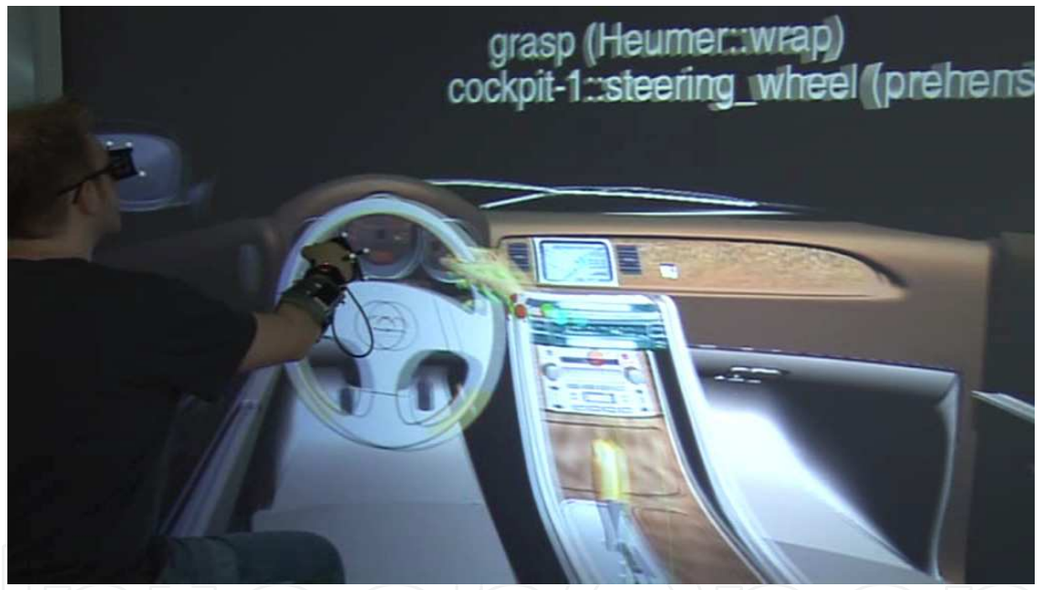


Fig. 9. Automatic recognition of basic interactions while manipulating the control actuators of a virtual car interior.

For full detail of the interaction, a complete history can be recorded (current state in every frame). This history is stored globally under a unique ID and is referenced by this ID in the interaction event fired at the end of the interaction. This enables the reproduction of either the outcome of the interaction (just final state), a reproduction of the state change order (intermediate discrete states) or an exact reproduction of the whole interaction (history). By the latter a replay in a purely graphical scene can be realized without the further need for an active dynamics simulation.

The result of the complete interaction analysis process is a stream of basic interactions which are persistently stored and also propagated to other application parts via the event system. There, they can be analyzed and processed further with automatic methods (action recognition and animation generation) or by hand.

4.4 Visual analysis tool

In order to monitor and improve the classification processes for interaction events (described in the previous section), we implemented a graphical tool that allows us to visualize, analyze and easily edit the recorded interaction data from a database or from local files. We implemented our tool VisuAnIA (Visualizing and Analyzing tool for InterAction data) by using a plug-in architecture, to easily extend the application and to provide different views, e. g. a 3D trajectory view or an animated hand view.

A graphical browser for the different channels (see section 4.1) offers a generic overview with a timeline (see fig. 10, left view). According to the layers in the interaction database (fig. 7) this view is separated into three levels. The top level contains the interaction events which are referring to the segmented hand trajectories in the second level, as well as to the raw data in the third level that has been captured from various input devices such as motion trackers, data gloves, etc. Following the FacetZoom metaphor as described in (Dachselt et al., 2008), the navigation through the three hierarchical levels can be done by clicking on a desired level, at which adjacent levels will be represented smaller and remain in context. The interval-based data (that contains a start and end time) is drawn in a box-like manner along the timeline and is enriched with generic textual information, such as the type of an interaction event or the exact time values. Furthermore the user can select one or more of these interval-boxes to analyze them by using special views. Similar to an audio editing tool, the current position of the cursor determines a time value that will be used in other plug-ins. To enable the analysis even for very short or long datasets, this view also provides a linear zooming option along the timeline.

For a more specific visualization of the hand trajectories we implemented another plug-in that provides a 3D view (fig. 10, middle view). Next to rotating and zooming, this view allows the examination of smaller parts of the trajectories. This is done by defining an interval in the timeline view. Only those parts of the trajectories are displayed which are inside the interval, what is useful in particular when a recorded action is very local and a hand trajectory occludes parts of itself. In order to illustrate the timing of the hand movement during the interaction, on the segmented trajectories a little sphere is shown. This sphere indicates the position of the hand according to the currently selected time value

A further view shows an animated 3D hand model to visualize the data from a dataglove (fig. 10, right view). Again using the timeline, the user can access the recorded postures of

```
<event type="actuator" start-time="11.1374" duration="0.2058" id="CAE-2">
    [...]
    <actuator-states object-id="cockpit-1::steering_wheel">
      <start-data>
        <state dof="rotational1" value="-0.0094"/>
        <discrete-state value="center"/>
      </start-data>
      <goal-data>
        <state dof="rotational1" value="3.3676"/>
        <discrete-state value="right"/>
      </goal-data>
    </actuator-states>
</event>
```

Listing 2. Code example of a control actuator event.

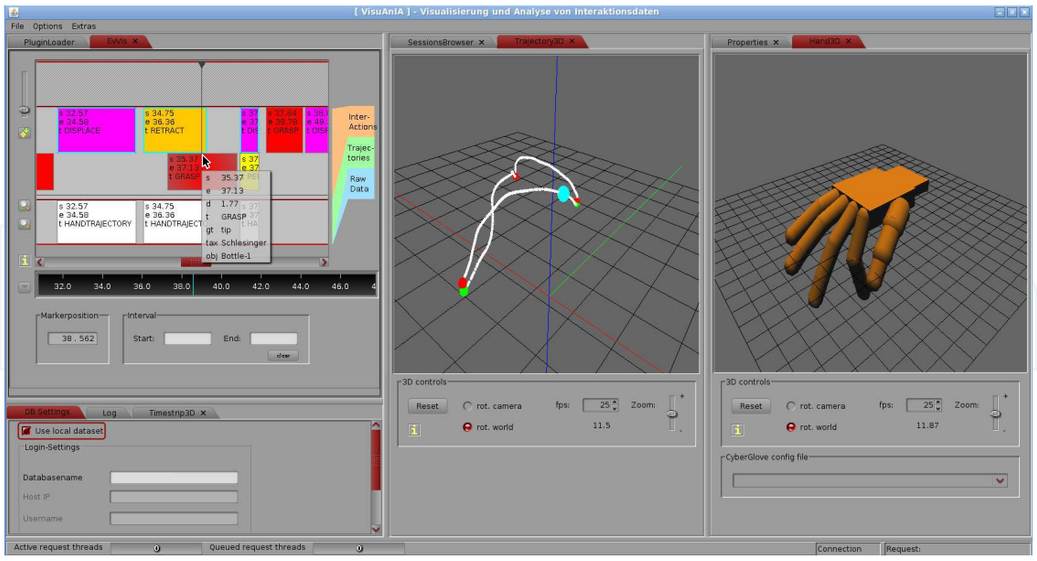


Fig. 10. VisuAnIA: The generic timeline view on the left is used to browse the data that gets visualized in special 3D views.

the hand and compare them to the types of the recognized interaction events or to the objects which have been used.

A plug-in with a commonly known generic properties view (fig. 11, right view) allows the listing of all the recorded values as they are stored in the database. This view also permits the modification of important attributes, such as timing values or the type of an interaction event. Due to the linear zooming option in the basic timeline view, all interval-boxes outside the focused area will disappear from the context so that large datasets may not be adequately examined. Therefore we implemented a conceptual 3D timeline plug-in (*Timestrip3D*) that only shows a small fraction from the basic timeline (fig. 11, lower view). To indicate the relation between the two timelines a small box, that defines the interval for the second timeline and marks the area of interest, is drawn in the basic timeline view. By using these two timelines it is possible to examine a focused area of the dataset without losing context. Additionally, the timestrip offers a fisheye like zooming option, that carries on the focus-plus-context representation and thus supports the user during analysis. The 3D strip actually consists of several Bézier surfaces that will be modified to achieve the zooming effect. Since the textual information on the timestrip is generated by using different font sizes, different zooming degrees implicate varying levels of detail of the presented information.

To keep both of the timelines synchronized, e. g. while using the generic timeline view to browse and analyze a dataset and the timestrip to obtain additional information, we implemented a new tool we call *Foculyzer*. The aim of the tool is to apply the currently examined (focused) area around the cursor from one timeline view automatically to a second timeline view. That means that a user does not need to set the area of interest (red box in fig. 11) by hand, as necessary e. g. in the TimeSearcher 2 tool (Buono et al., 2005). The tolerance range in which the cursor can be moved without shifting the area of interest can be freely adjusted in our tool. Thus, e. g. depending on the kind of analysis or length of the dataset, the user can customize the tool according to the current needs. Two sliders on the focused area are provided for such adjustments (fig. 11). The sliders are hidden during analysis so that the timeline itself won't be occluded. With our tool we overcame the two drawbacks: The first one occurs when the area of interest is centered exactly around the cursor, which means

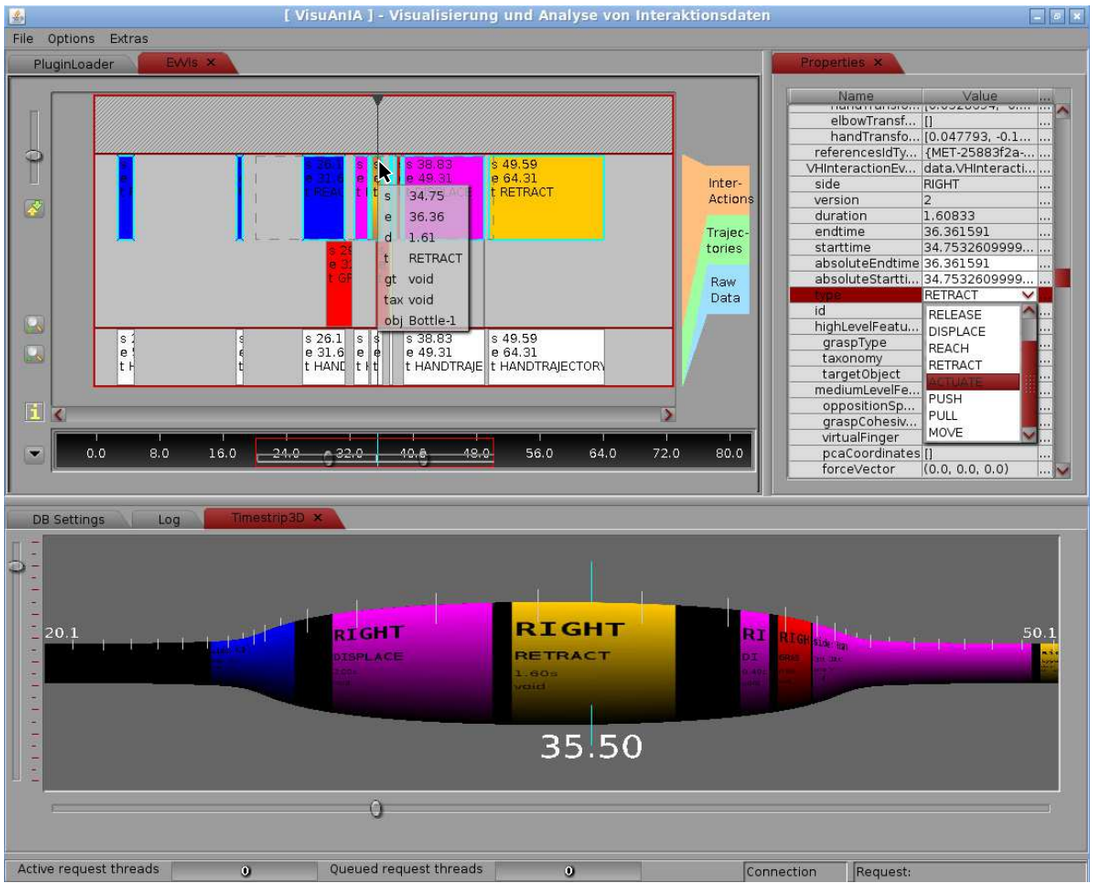


Fig. 11. While the user analyzes interaction events in the upper timeline view, the focused area is automatically applied to the timestrip where more information and a fisheye like zooming is provided. The range of tolerance for the cursor is controlled by the sliders on the red box. On the right, a generic properties view shows the values of the selected interval-boxes and allows the modification of important values.

that every little movement of the cursor causes the area to shift as described in (Bederson, 2000). A second drawback found in some systems results, when the area of interest is shifted only upon the collision of the cursor with the area bounds. Thus, in order to effect a focus change the cursor has to be moved to the borders explicitly which can be annoying. The adjustable tolerance range allows for a more flexible interaction instead. Our tool Foculyzer is not restricted to interaction events, but can also be used to synchronize various other views, e. g. in audio or video editing programs to analyze different channels or different views of one channel.
Future versions of our tool VisuAnIA will provide plug-ins for data filtering, e. g. to recognize wrongly tracked trajectories. Furthermore, we will implement a player to reproduce the recorded interaction data in various speeds.

5. Application scenarios

Due to its universal and extendable nature, the framework for control actuators presented here can be used in many different types of interactive VR prototyping applications where

movable parts with physically believable behavior are called for. In our particular area of research - which has virtual prototyping as application domain - we use the framework to simulate parts of machinery or other virtual prototypes. The operation is done manually by a VR user. The aim is to demonstrate manipulation sequences that are repeated by an animated character, i.e. a virtual human, see Figure 12. This type of animation is closely related to *Programming by Demonstration* in the field of robotics and is referred to as *Action Capture* in (Jung et al., 2006).

Fig. 12. Control actuators contained in the virtual prototype of a car cockpit.

Apart from our research scenario many different use cases are conceivable. Such applications can include virtual prototyping and construction, ergonomics studies, instructional animations, etc. The only prerequisite is a mechanism to determine forces and torques exerted on the actuators. A simple mechanism would be to translate 6DOF input from e. g. a Spaceball or a tracking device and translate its output to appropriate forces and torques.
Furthermore, interactive scene functionality can be realized by subscribing and reacting to interaction events. For example, a car radio sound can be played when the car radio button is pressed. The volume can be adjusted, when the knob is turned, etc.
In this section, we discuss the benefits of our approach for two typical application domains (virtual training, ergonomic studies with virtual prototypes) on the basis of two scenarios we have (partially) implemented.

Scenarios

The first scenario is a control panel on which different actuators are located (see fig. 1). These VR objects represent reference implementations of the fundamental types of control actuators specified in (DIN894-3, 2006). Among these control actuators are a steering wheel, different knobs and sliders.
The second scenario is a car driving application (fig. 13). In this scenario, the user sits in a virtual cockpit and manipulates control actuators in order to drive the car. Control actuators are a steering wheel, a gear shift, radio knobs and the ignition key. The user's interactions trigger certain functionalities and animations.

Fig. 13. A user manipulates the gear shift in our car driving scenario.

Virtual training and functional prototypes

An important use case of our approach is virtual training. Our first prototype serves the purpose to improve the user's fundamental skills in operating virtual control actuators. The user can change the state of an actuator in order to explore its behavior.

In the second scenario, the user can learn how to start and drive a car or he can improve his driving skills. By connecting the manipulation of an actuator with a certain functionality using the event mechanism described before, functional prototypes can be realized. For example, the user can hear the sound of the car's engine after starting it and while driving; he can hear music from the radio after turning it on and he can navigate the car through the virtual environment by rotating the steering wheel. By providing physically correct behavior of actuators in combination with functional prototypes, the degree of realism and the feeling of immersion can be dramatically improved. In this way, the training scenario becomes a very realistic user experience which facilitates the transfer of learned skills from a virtual to a real environment.

Ergonomic studies

A main application domain of our approach is ergonomic evaluation of virtual prototypes. The evaluations can be conducted with real VR users and virtual humans. An example of a virtual prototype suitable for ergonomic studies is a virtual car as described in the second scenario. The car prototype provides a set of actuators which are typically required for driving. A user performs different actions, such as turning the ignition key, turning the radio on, switching the gears and turning the steering wheel. These actions can be imitated by virtual humans with different body proportions. Ergonomic problems occur, e. g., if a (virtual) human's arm collides with an obstacle while trying to grasp a target object or if the human cannot even reach the target object. Such problems can be identified and analyzed by observing the real VR user performing the actions required to reach a certain goal (e. g. starting

the car's engine) or by watching animations of virtual humans synthesized from captured action sequences. In this context, realistic actuator behavior is important, since a state change of an actuator can influence ergonomic conditions. For instance, if there were no collisions with the gear shift lever, such collisions can suddenly occur after changing the lever's state (orientation/position). In addition, by using an event serialization mechanism, it is not only possible to reproduce the motions of the involved control actuators exactly (even without virtual humans), but also to visualize the generated events for a more thorough analysis (e. g., in order to identify unwanted collisions). This is an important use case for our VisuAnIA tool (see 4.4) that offers the required analysis functionality.

6. Conclusion

We presented a comprehensive approach for enabling and analyzing natural interactions with functional virtual prototypes. Compared to more generic languages such as X3D or Collada, a domain-specific XML-based language simplifies the specification of virtual control actuators. In immersive VR settings, control actuators can be manipulated using natural interaction techniques based on data gloves and motion tracking devices. During on-going user interactions, the current state of the control actuator is continuously monitored and made available to other components of the overall VR-system by means of an event mechanism. A continuously moving control actuator may snap to predefined discrete states, e. g. on/off positions of a switch, and special events are fired when the actuator reaches such a discrete state. For instance, this allows to simulate functional effects of control actuators in the VR system. The described approach covers all control actuators described in the European standard EN 894-3.

True functional validation of virtual prototypes, in our opinion, should include but not be restricted to live experiences of VR users. Therefore, our approach comprises several concepts and tools for recording and offline analysis of operating procedures performed on virtual prototypes in immersive VR. For this, various aspects of user interactions can be stored in an interaction database, from low-level sensor data, over trajectory data, up to high-level interaction events containing e. g. the classifications of user grasps w.r.t. a grasp taxonomy. A graphical tool that builds on state-of-the-art as well as newly designed information visualization methods serves to analyze and, if appropriate, edit the recorded interactions. The power of our approach is demonstrated by an implemented application where animations of virtual humans are generated from the recorded interaction data. We believe that several application settings, such as VR-based training systems, could benefit from the presented approach.

7. References

Abate, A. F., Guida, M., Leoncini, P., Nappi, M. & Ricciardi, S. (2009). A Haptic-Based Approach to Virtual Training for Aerospace Industry, *J. Vis. Lang. Comput.* 20: 318–325.
URL: *http://portal.acm.org/citation.cfm?id=1598095.1598605*

Bederson, B. B. (2000). Fisheye Menus, *Proceedings of the 13th annual ACM symposium on User interface software and technology*, UIST '00, ACM, New York, NY, USA, pp. 217–225.
URL: *http://doi.acm.org/10.1145/354401.354782*

Buono, P., Aris, A., Plaisant, C., Khella, A. & Shneiderman, B. (2005). Interactive Pattern Search in Time Series, Vol. 5669, SPIE, pp. 175–186.
URL: *http://link.aip.org/link/?PSI/5669/175/1*

Cutkosky, M. (1989). On Grasp Choice, Grasp Models and the Design of Hands for Manufacturing Tasks, *IEEE Transactions on Robotics and Automation* 5(3): 269–279.

Dachselt, R., Frisch, M. & Weiland, M. (2008). Facetzoom: A Continuous Multi-Scale Widget for Navigating Hierarchical Metadata, *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, ACM, New York, NY, USA, pp. 1353–1356.

DIN894-3 (2006). Safety of Machinery - Ergonomic Requirements for the Design of Displays and Control Actuators - part 3: Control Actuators, en 894-3.

Heumer, G. (2010). *Simulation, Erfassung und Analyse direkter Objektmanipulationen in virtuellen Umgebungen*, PhD thesis, TU BA Freiberg.
URL: *http://nbn-resolving.de/urn:nbn:de:bsz:105-qucosa-70518*

Heumer, G., Ben Amor, H. & Jung, B. (2008). Grasp Recognition for Uncalibrated Data Gloves: A Machine Learning Approach, *Presence: Teleoperators and Virtual Environments* 17: 121–142.
URL: *http://portal.acm.org/citation.cfm?id=1362509.1362512*

International Organization for Standardization (2008). *ISO/IEC FDIS 19775-1:2008: Information technology – Computer graphics and image processing – Extensible 3D (X3D) – Part 1: Architecture and base components, 2. Edition*.

Jerez, J. & Suero, A. (2011). Newton Game Dynamics. Open-Source Physics Engine .
URL: *http://newtondynamics.com*

Jung, B., Amor, H. B., Heumer, G. & Weber, M. (2006). From Motion Capture to Action Capture: A Review of Imitation Learning Techniques and their Application to VR-based Character Animation, *Proceedings VRST 2006 - Thirteenth ACM Symposium on Virtual Reality Software and Technology*, pp. 145–154.

Kallmann, M. & Thalmann, D. (1999). Direct 3D Interaction with Smart Objects, *Proceedings ACM VRST 99, London*.

Khronos Group (2008). *Collada - Digital Asset Schema Release 1.5.0 Specification*.

Moehring, M. & Froehlich, B. (2011). Natural Interaction Metaphors for Functional Validations of Virtual Car Models, *IEEE Transactions on Visualization and Computer Graphics (TVCG), accepted manuscript* .
URL: *http://dx.doi.org/10.1109/TVCG.2011.36*

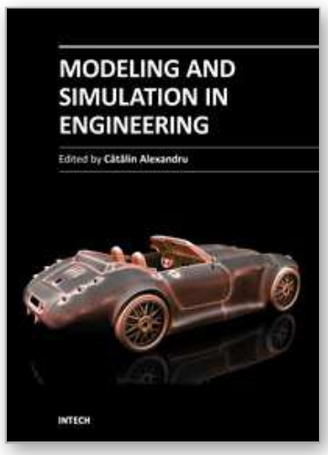
Moehring, M. & Fröhlich, B. (2010). Enabling Functional Validation of Virtual Cars Through Natural Interaction Metaphors, *Proceedings of IEEE Virtual Reality Conference, VR 2010*, pp. 27–34.

Schlesinger, G. (1919). Der Mechanische Aufbau der Künstlichen Glieder, *in* M. Borchardt et al. (eds), *Ersatzglieder und Arbeitshilfen für Kriegsbeschädigte und Unfallverletzte*, Springer-Verlag, Berlin, Germany, pp. 321–661.

Weber, M., Heumer, G., Amor, H. B. & Jung, B. (2006). An Animation System for Imitation of Object Grasping in Virtual Reality, *Proceedings of Advances in Artificial Reality and Tele-Existence, 16th International Conference on Artificial Reality and Telexistence, ICAT*, Springer, pp. 65–76.

Zachmann, G. & Rettig, A. (2001). Natural and Robust Interaction in Virtual Assembly Simulation , *Eighth ISPE International Conference on Concurrent Engineering: Research and Applications (ISPE/CE2001)*, pp. 425–434.

Zorriassatine, F., Wykes, C., Parkin, R. & Gindy, N. (2003). A Survey of Virtual Prototyping Techniques for Mechanical Product Development, *Journal of Engineering Manufacture* 217.



Modeling and Simulation in Engineering
Edited by Prof. Catalin Alexandru

ISBN 978-953-51-0012-6
Hard cover, 298 pages
Publisher InTech
Published online 07, March, 2012
Published in print edition March, 2012

This book provides an open platform to establish and share knowledge developed by scholars, scientists, and engineers from all over the world, about various applications of the modeling and simulation in the design process of products, in various engineering fields. The book consists of 12 chapters arranged in two sections (3D Modeling and Virtual Prototyping), reflecting the multidimensionality of applications related to modeling and simulation. Some of the most recent modeling and simulation techniques, as well as some of the most accurate and sophisticated software in treating complex systems, are applied. All the original contributions in this book are jointed by the basic principle of a successful modeling and simulation process: as complex as necessary, and as simple as possible. The idea is to manipulate the simplifying assumptions in a way that reduces the complexity of the model (in order to make a real-time simulation), but without altering the precision of the results.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen