Abstract. We present a simple concept for the 3D-visualization of systems that are modelled as a Petri net. To this end, the Petri net is equipped with some information on the physical objects corresponding to the tokens on the places. Moreover, we discuss a prototype of a tool implementing this concept: PNVis version 0.8.0.

Keywords: Petri nets, 3D-visualization, animation.

1 Introduction

Petri nets are a well-accepted formalism for modelling concurrent and distributed systems in various application areas: Workflow management, embedded systems, production systems, and traffic control are but a few examples. The main advantages of Petri nets are their graphical notation, their simple semantics, and the rich theory for analyzing their behaviour.

In spite of their graphical nature, getting an understanding of a complex system just from studying the Petri net model itself is quite hard – if not impossible. In particular, this applies to experts from some application area who, typically, are not experts in Petri nets. ‘Playing the token-game’ is not enough for understanding the behaviour of a complex system. Using suggestive icons for transitions and places of the Petri net in order to indicate the corresponding action or document in the application area is only a first step.

Therefore, there have been different approaches that try to visualize the behaviour of a Petri net in a way understandable for experts in the application area. At best, there will be an animation of the model using icons and graphical features from the corresponding application area. ExSpect [14], for example, uses the concept of a dashboard in order to visualize the dynamic behaviour of a system in a way that is familiar to the experts in the application area (e.g. by using flow meters, flashing lights, etc. as used in typical control panels). In ExSpect, it is even possible to interact with the simulation via this dashboard. Another example is the Mimic library of Design/CPN [4, 13], which allows a Design/CPN simulation to manipulate graphical objects, and the user can interact with the simulation via these graphical elements. This way, one can get a good impression of the ‘look and feel’ of the final product. A good example is the model of a mobile phone [10]. Another approach for visualizing Petri nets is based on graph transformations and their animation: GenGED [1, 2].
In the PNVis project, we take the next step: The graphical objects manipulated by the simulation are no longer considered to be artifacts for visualizing information; rather we consider them as a part of the system model. Actually, they are considered as the physical part of the system. Though simple, this step has several benefits: First, it makes the interaction between the control system and the physical world explicit. Second, the physical part can be used for a realistic 3D-visualization of the dynamic behaviour by using the shape and the dynamic properties of the physical components. Third, the properties of the physical objects can be used for analysis and verification purposes. For example, we can exploit the fact that two physical components cannot be at the same place at a time.

In this paper, we show how a Petri net can be equipped with the information on the physical objects. Moreover, we discuss the concepts and a prototype of a tool that uses this information for a 3D-visualization of the system: PNVis version 0.8.0. Here, the focus is on those aspects of the physical objects that are necessary for visualization: basically the shape of the objects. The prototype of PNVis is restricted to low-level Petri nets. The PNVis project, however, has a much wider scope. For example, we would like to use some physical properties of the objects such as their weight for analysis purposes. Moreover, PNVis will support high-level Petri nets, and it will provide concepts for constructing a system from components in a hierarchical way.

PEP [11] was one of the first Petri Net tools that came up with a 3D-visualization of Petri net models: SimPep [6]. Basically, SimPep triggers animations in a VRML model [5] while simulating the underlying Petri net. But, this simulation imposes a severe restriction on the animations: There is only one animation at a time; concurrent animations of independent objects are impossible. In this paper, we will present concepts that allow us to have concurrent animations of independent objects. The trick is to associate animations with places rather than with transitions.

2 Concepts

In order to animate the behaviour of a Petri net in a 3D-visualization, the net must be equipped with some information on the physical objects. Moreover, the behaviour of the objects must be related to the dynamic behaviour of the Petri net. In the following, we will discuss how to add this information to a net.

Geometry, shapes and animation functions. In a first step, we distinguish those places of a Petri net that correspond to physical objects. We call these animation places. The idea is that each token on an animation place corresponds to a physical object with its individual appearance and behaviour. In order to animate a physical object, we need two pieces of information: its shape and its behaviour.

Defining the shape of the object is easy: Each animation place is associated with a 3D-model (e.g. a VRML model [5]) that defines the shape of all tokens on this place. Defining the behaviour of an object is similar: Each animation