Mining Simulation

Multipurpose models for large–scale systems

An internship report

by Felix Ritter

"Otto von Guericke" University Magdeburg

June 9, 1998

Acknowledgments

I wish to thank John R. Sturgul, Ph.D. and his wife Alison for the great help and support during my stay at the University of Idaho, where the fundamental work for this paper took place. Thanks are also due to Prof. Peter Lorenz for his assistance in organizational questions and his helpful hints regarding this paper.

Felix Ritter

Preface

Nowadays, the use of simulations is paid much more attention than a few years ago. Particularly in big industries, simulators are successfully introduced and accepted as a way to evaluate design alternatives and to validate decision processes. Simulations assist in project design as well as in optimization of existing production facilities. Planned investments can be done directly, effectively and with less risk.

Obviously, the advantages of simulation are indisputable and the simulators are object of an ongoing development. However, the common modeling methods lag far behind as one model can only be used for one project. This is a luxury no business can afford today. The reusability of already obtained know–how is a requirement to react quickly to the market and thus to be flexible. Particularly reusability, however, tends to be neglected when considering simulation models.

This paper is especially concerned with multipurpose simulation for large–scale systems whereby special emphasis is placed on mining simulation. The practical examples and concepts outlined within the paper mostly derive from a mine simulation project performed for the Kennecott Corporation, Utah, USA. To underpin these approaches, multipurpose modeling concepts as well as modern concepts for large–scale systems are further subjects to be dealt with. Additional topics describe the visualization and representation of simulated processes and the integration of continuous and discrete simulation approaches. In conclusion, details of the mine project are explained more closely.

Chapter 1 discusses fundamental concepts and methods facilitating the design of multipurpose models. It also gives a short introduction to the High Level Architecture.

Chapter 2 describes design approaches to simulation mainly concerned with modular modeling. This chapter continues in part the concepts of the previous chapter. Further topics deal with the integration of field–specific knowledge in the simulation process and with Distributed Interactive Simulation.

Chapter 3 is concerned with several aspects related to the visualization of simulation processes and results. It investigates different approaches and systems based on them. In addition, it gives some hints regarding the adaptation

of existing CAD layouts. As another subject, it suggests some ways to optimize facility layouts and to visualize this process.

Chapter 4 focuses on the adaptation of continuous approaches to discrete simulation with regard to modeling and visualization. It explains the characteristics of both concepts and discusses the problems arising from a combination.

Chapter 5 finally describes the simulation project *Kennecott's Greens Creek Mine* in general and focuses on particular problems and solutions, which then get explained more detailed. Due to the split of this paper[1], only the subsurface and the ore–mill are covered.

[1]This paper is split into two reports. For further information about this project, such as the surface and the dock area, see [Göt98].

Contents

Preface **iii**

1 Multipurpose modeling concepts **1**
1.1 General considerations . . . 2
1.2 Abstract procedure . . . 2
1.3 Reusable models according to HLA . . . 5

2 Design concepts for large–scale systems **7**
2.1 Modular simulation models . . . 7
Modularization of the simulated system . . . 8
Interface design . . . 10
Workgroup–orientated modeling . . . 11
2.2 Knowledge–based simulations . . . 12
2.3 Distributed simulation . . . 14

3 Visualization of simulations **16**
3.1 Visualization by tables and diagrams . . . 16
3.2 Different approaches for animations . . . 17
Graphical simulation systems with object library . . . 18
Classical simulation systems with graphical extensions . . . 20
Virtual Reality and interactive simulation systems . . . 25
3.3 Adaptation of existing CAD layouts . . . 27
3.4 Layout optimization and its visualization . . . 28

4 Continuous approaches in discrete simulation systems **30**
4.1 Different approaches to modeling . . . 30
4.2 Adaptation of continuous processes . . . 31
4.3 Visualization of continuous processes . . . 32

5 Mining simulation with GPSS/H and Proof–Animation **34**
5.1 Subsurface area . . . 34
Motivation . . . 34
General structure . . . 36
Simulation details . . . 39

5.2 Ore mill 44
Motivation 44
General structure 44
Simulation details 46

A Screenshots of Kennecott's Greens Creek Mine simulation 50

Bibliography 53

List of Figures

1.1 Abstract approach to build a multipurpose model 3
1.2 Functional view of the HLA architecture 6

2.1 The components and connections of a simple mine model 8
2.2 Knowledge–based simulation architecture 13

3.1 The graphical simulation system Quest 19
3.2 Combination of Virtual Reality and simulation 25
3.3 SOS–VR — Self–Organization Simulation and Virtual Reality . . 27

4.1 Example of a discrete and continuous simulation approach 31
4.2 Discrete approximation of a continuous function 32
4.3 Continuous simulation results expressed by curves 33

5.1 The main components of the subsurface area 35
5.2 Layout detail showing the subsurface area of the mine 37
5.3 The different types of passing bays 38
5.4 Abstract plan of the tunnel system 43
5.5 Flowchart of the milling process 45

A.1 Layout detail depicting the working cycle of an ore truck 51
A.2 The different layouts of the ore mill 52

Chapter 1

Multipurpose modeling concepts

Before we start to discuss multipurpose simulations, let us visualize the common procedure to perform a simulation for an actual project.

1) A problem arises and shall be solved with the help of a simulation expert. The person posing the problem is usually not very familiar with simulation as such and therefore has no concrete idea what he can expect and how long it will take to be realized. Thus, it is up to the modeler to develop an abstract plan for that simulation.

2) The simulation model has to be build. This is the most complex and time consuming process and needs an intensive communication between the one who knows the problem in detail and the modeler. Furthermore, that period is characterized by lots of changes partially due to insufficient specification of some components which should be included. Quite often, the estimated time for that purpose will be exceeded.

3) The simulation model has to be validated. Test data shall prove if the model behaves like the real system or is at least similar. The simulation expert is going to present the model and the results of the simulation to the customer. The latter one gives him to understand that he really needs the results urgently, and that the time to develop the model is over anyway. However, during the presentation it often turnes out that the modeler had been partly mistaken in the problem or something is not satisfying the client. Probably, the real system was altered too, and thus the model is not up to date. The modeler remains with only little time to do the changes in the simulation model.

4) With the changed and proven model will be performed some simulation experiments. The client discloses that he has to order the equipment urgently, and he needs the results now otherwise they will be without use for him. The modeler continually sends partial results.

5) The final results, which lost their significance in the meantime, will be presented to the client again. He thanks the simulation expert for his efforts

and thinks about the possibility of some changes in the running installation process due to the outcomes of the simulation.

6) The simulation model will be saved on tape, stored in a safe and forgotten.

This sounds hard, but it shows how important flexibility and prompt realization are for custom–tailored simulation projects. A flexible design of the simulation model lays the foundations for adaptability and reusability. This chapter will discuss some approaches to design simulations which can be used for more than only one project.

1.1 General considerations

In a special kind of business, many systems, such as plants and facilities, are quite similar in their key features or in the points which are interesting to analyze. Therefore, it could save a significant amount of time to have a model that allows one to examine the problems of most of these systems.

Initially, in many respects, it is much easier to design a model for a single project rather than a multipurpose one, which, regarding the questions the specific model should help to answer, requires much more effort. Some years ago, the lower performance of computers did also prevent the writing of complex programs such as multipurpose simulations for large–scale systems. In fact, the overhead these models have and need to have to support the multipurpose aspect is considerable compared with the capabilities of these computers. Today, however, there is no reason to refuse this anymore.

In [Wal94], for instance, the author describes a model to investigate quite different systems such as the train traffic and the telephone networks. Although they are quite dissimilar at first sight, the structural characteristics are rather the same. Both systems have a network like infrastructure, they need to make routes for the transport and they share some of the operating problems as well.

Obviously, the model above is an extreme but also a good example for multipurpose simulations. On closer inspection, even quite different systems may have a great deal in common. This applies to mining as well. Even though they are found in different mine systems, the main facilities such as the ore mill, backfill plant, stock piles and the pit system itself are very much alike. Thus, one needs to find a common level allowing the sufficient examination for more than only one system.

1.2 Abstract procedure

At the beginning, the intended purpose for the simulation has to be defined since even the most abstract multipurpose simulation model will not fit all systems.

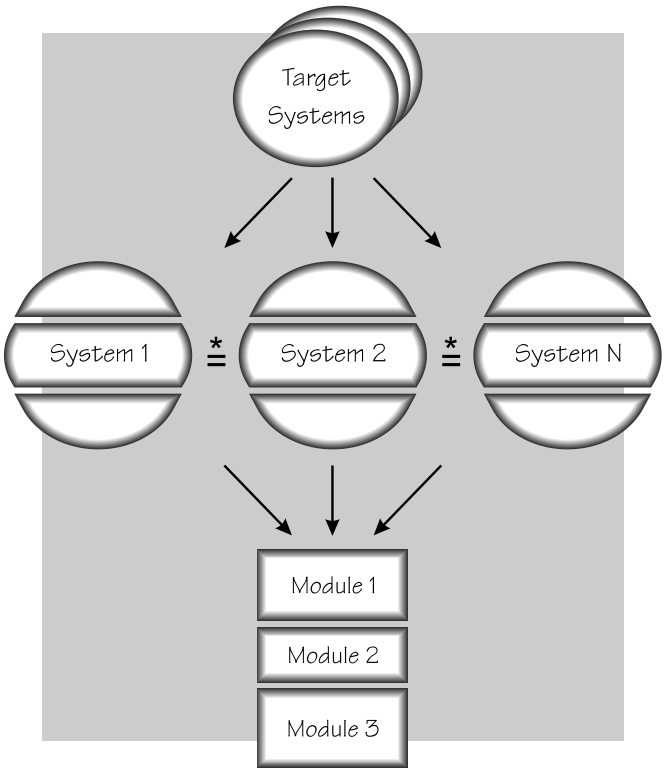


Figure 1.1: Abstract approach to build a multipurpose model considering the key characteristics of the target systems

If the model is to cover a great deal of systems, it might be necessary to reduce the level of detail, whereas systems with need for precise analysis require more detailed modeling. Hence, the model will only be a compromise between the number of systems that can be investigated and the coverage of special features.

To determine and to define the operation field for the simulation, it is best to make a list of features the multipurpose model should provide. By investigating the target systems[1] regarding the key features, characteristics and their relations not only the important components will be found but also the structure the model can be build of. To sum it up, the following steps may be carried out (see also Figure 1.1):

1. Define the overall operation field of the multipurpose model.

2. Investigate the target systems with regard to their components and key features.

[1] The term *target systems* stands for an imaginable set of systems that operate in a common manner and that could be taken as representatives for the operation field.

3. Compare these components considering the components the model should consist of.

4. Develop the main structure of the multipurpose model according to the connections among the components.

5. Design the model with regard to reusability and flexibility.

Some of these steps (particularly the first three) may be performed in a slightly different order. It might be also helpful, depending on the dynamics of the operation field, to investigate some target systems first to get a clear idea of what the operation field should be.

The main structure represents the connections between the entities the modeler has chosen to subdivide the overall system. Intuitively, these will be relatively self–contained facilities that most of the target systems contain. As more components are modeled independently, less problems arise if changes have to be applied since the information flow between them has been minimized and with it the interdependencies. The subdivision of the model also provides more flexibility for the modeler and simplifies the maintenance as well. With it, however, the interface among these modules respective objects gains in importance. It has to be flexible enough to cover not only the current but also future purposes. In the beginning, a discussion on this subject helps to prevent gaps. Section 2.1 shows some strategies to build a subdivided model.

As mentioned, in mining, most of the mines are based on a general structure which also implies similar facilities. Thus, a model containing the main components, such as a pit system, ore mill and stock piles might be a reasonable starting point to further design a multipurpose model for mine simulation. Depending on the desired level of abstraction, some of these components need to be subdivided and modeled in more detail. For a precise analysis of the pit system itself it is significant if it is an open–pit mine or a subsurface one. On the other hand, investigations of the mill, for instance, only require an abstract source from where the ore comes. To give another example, the model for a pit system could be designed completely independent of the ore which is mined and which the trucks carry. However, the chemical processes in the mill are quite different concerning the ore type. Above all, to satisfy the claims a more or less complex model has to be designed and the complexity is highly dependent on these claims.

Although the mine model explained in Chapter 5 was not intended to investigate several mines, some of its components are self–contained enough to facilitate reuse. At the lowest subdivision level two components can be distinguished. The subsurface and surface area were designed relatively independently. Obviously, it is hard to include these components in another model without significant changes, since at this level models differ most and all specialties, in which the model behaves uniquely, can be found there. Hence, the components need to be further subdivided. The mill, however, can be used quite universally because its design is

not so special and the parameters were defined in an separate data file. Thus they can be changed easily. All it needs is a source of crude ore and destinations for the processed ingredients. Like this mine model, many simulation models were build upon a quite flexible parameter set, because if they would not allow to be modified in their behavior not much conclusions could be drawn. Indeed, some models can be used to examine other systems without much changes.

1.3 Reusable models according to HLA

A multipurpose model should cover a certain number of systems without the need of internal modifications. However, a more specific simulation model that, nonetheless, promotes the reuse of its components without many changes of the structure represents a multipurpose model as well.

The *American Department of Defense* (DoD) has developed the *High Level Architecture* (HLA [HLA96]) for modeling and simulation. HLA is the designated successor of DIS (see also Section 2.3). The purpose of this architecture is to facilitate interoperability among simulations and the aforementioned reuse of the simulation and its model.

The HLA is based on the premise that no single model or simulation can satisfy all uses and users at all levels of resolution. An individual simulation or set of simulations developed for one purpose can be applied to another application under the HLA concept of the federation[2].

The HLA does require that each simulation and federation document its object model using a standard Object Model Template (OMT). These templates are intended to be the means for open information sharing across the community to facilitate reuse of simulations. With the use of a template, it is possible to develop tools which allow for automated search and reasoning about object model template data, to further facilitate cost–effective information exchange and reuse.

The backbone of an HLA federation is the Runtime Infrastructure (RTI) which is, in effect, a distributed operating system for federations. It is implemented in such a way that it is broadly applicable and reusable across a wide range of modeling and simulation applications. It provides a set of basic services to the federation, which interoperates via the RTI (Figure 1.2).

One of the goals of the HLA development process is to make as many reusable as possible, but with open specifications which will allow for innovation and adaptation. The HLA does not impose the internal system architecture of a federate[3], but rather addresses the manner in which the federate operates with the RTI and with other federates.

[2] A named set of interacting federates, a common federation object model, and supporting RTI, that are used as a whole to archive some specific objective.

[3] A member of a HLA federation. All applications participating in a federation are called *federates*. In reality, this may include Federate Managers, data collectors, live entity surrogates simulations, or passive viewers.

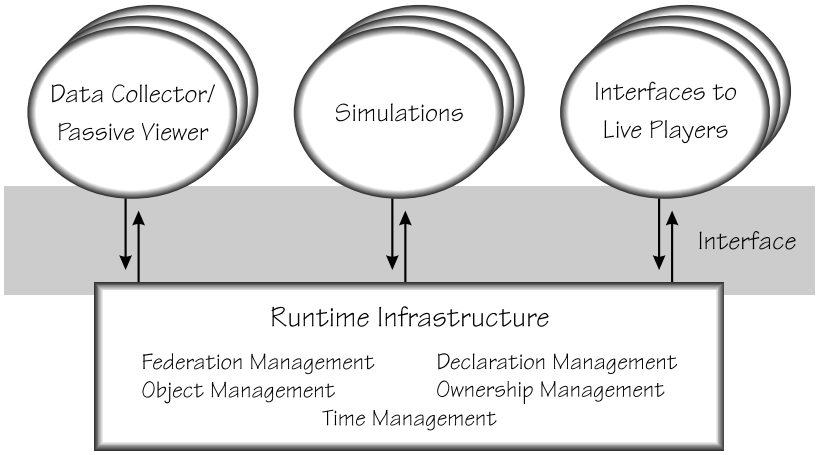


Figure 1.2: Functional view of the HLA architecture

Beyond this, the HLA framework also provides the structural basis for interoperability. Most of the rules appearing in the definition have been included for that reason. Thus, the intercommunication between simulations and applications supporting special operations obtains an increased quality with HLA. Complete new aspects are offered to the simulation and simulation application developers, as they are able to design tools that are applicable to more than one simulation system.

The HLA is defined by three documents:

- The HLA Rules — general principles of the HLA applying to the federates
- The HLA Interface Specification — between federates and the Runtime Infrastructure (RTI)
- The HLA Object Model Template (OMT) — for documenting key information about simulations and federations

This is a serious effort to provide a specification for further modeling and simulation. By it, it will be possible to reuse an existing model for a further one and thus to ultimately reduce the cost and time required to create an entirely new system for a similar purpose.

Chapter 2

Design concepts for large–scale systems

Today's modeler are faced with the necessity to write ever increasing simulations in shorter time periods and to expand existing models as they represent system changes. Hence, modern concepts in simulation, especially for large–scale models, are particularly concerned with the following subjects (which do not claim to be complete):

- Reuse and adaptation of existing models for other purposes, which include the adaptation of a model to changes in the represented system
- Integration of data bases and expert systems to assist in model design and in the determination of simulation parameters
- Distributed simulations, allowing world wide cross–linked simulations
- Visualization of simulations to facilitate the interpretation of simulation results as well as the discovery of irregularities

The following sections deal with these subjects or describe approaches to meet their requirements. The topic *Visualization of simulations* will be separately discussed in Chapter 3.

2.1 Modular simulation models

Large–scale simulations are often written by a team. Thus the distributed work requires definitions to ensure interoperability. A modular model with documented coherent interfaces bears not only a high potential of reuse but further decreases the need of coordination within the design process. Moreover, it provides a fast way to react to changes since modules can be altered without many impacts on the underlying model structure.

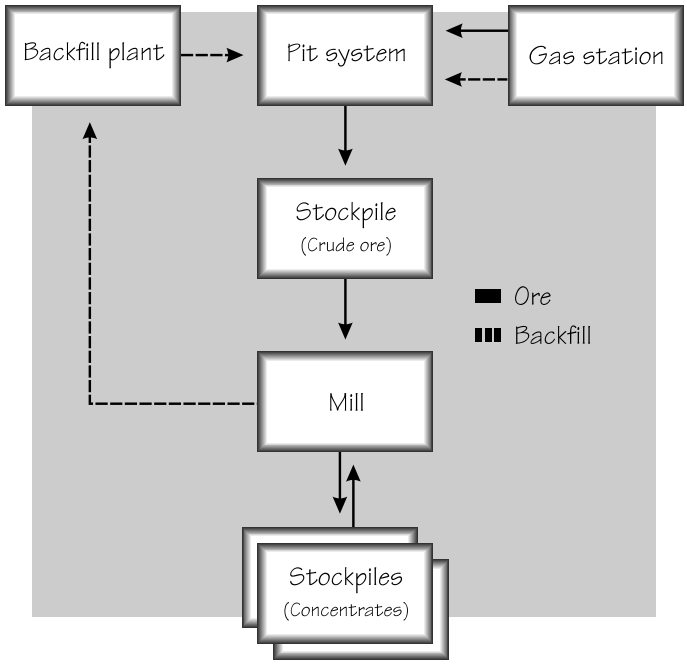


Figure 2.1: The components of a simple mine model including their connections, distinguished between ore and backfill related dependencies (→ ≡ impacts on)

Modularization of the simulated system

To get some assistance in modular modeling, it might be helpful to have a drawing that visualizes the main components and their connections. Flow charts are common to represent the different stages within the working cycle (Figure 5.5, e.g., depicts the processing of the crude ore by an ore mill). Often, however, the modeler only needs a rough sketch to keep track of the dependencies among modules or to discuss the ongoing development process of the simulation model with the customer in the early stages (Figure 2.1).

A module[1] should consist of a relatively self–contained unit, such as a production facility. By the reduction of the dependencies on other modules the interface design becomes more simple and the connections are far more obvious. With regard to the reusability of the whole model, modules should be exchangeable with ones offering similar functionality. In a mine model, for instance, the pit system could be an open–pit one as well as a subsurface one. If the minings would be replaceable, a great deal of flexibility could be applied to the entire model.

Chapter 5 describes the simulation of a complex subsurface mine, written in GPSS/H, which consists of 3 modules: the ore mill, the remaining surface plant and the underground minings. The model promotes the reuse of these modules

[1] Modularity is not bound to an object oriented simulation system allowing for inheritance etc., which is also the reason for speaking about modules instead of objects.

as they can be replaced to form a new model, such as the ore mill with another mill and the subsurface with other open–pit mines. However, to model efficiently, the modules need to be further subdivided. Therefore, the underground mines have been implemented as modules too. Since the modules contain a defined interface, modifications do not affect the data exchange as long as they conform to the interface specification. Thus, the actual working cycle in the mining areas can be altered without changes in the underlying structure.

Usually, the modeler not only wants to virtually subdivide the model but furthermore to separate the modules into different files. This is quite handy and refers to our more or less marked urge for order. In GPSS/H, modules can be shared and distributed among different files. The *INSERT* instruction allows to include code at arbitrary place. However, the modeler has to struggle with two limitations of GPSS/H. The names of model elements, such as queues, facilities etc., must be not longer than 8 characters, which makes it difficult to define module–intern variables even labels, in particular as they cannot be hidden from the code outside the module. An alternative is to implement the logic as macro and to "cipher" these elements at the time the model code gets translated.

```
 1  MODULE1  STARTMACRO #A,#B
 2
 3           GATE  FV   DEVICE#A,*+4     // facility is available
 4
 5           SEIZE      DEVICE#A         // seizes the facility
 6           ADVANCE    #B               // time for whatever has to be done
 7           RELEASE    DEVICE#A         // releases the facility
 8
 9           ENDMACRO
10
11
12           ...
13  MODULE1  MACRO 01,3.75
14           ...
15  MODULE1  MACRO 02,4.25
16           ...
```

The macro in line 13 expands the facility name to `DEVICE01` whereas the one in line 15 results in `DEVICE02`. Nonetheless, the modeler has to make sure that both facility names are unique in the entire model code.

Macros bear a high potential of reuse because they are naturally written to serve in multiple places and often for slightly different purposes as well. By extraction and generalization of code (variable parameters, common interface), modules can be built and used like modules of ordinary object libraries.

To sum up, modules are necessary to hold the design efforts on a low level and to provide an reliable way to change the behavior of the model. Furthermore, they help to reuse written code and with it already invested time.

Interface design

The interface design plays an important role within the development of the simulation model. Since multipurpose models claim to provide reusable code, each module has to specify how it connects to the model structure, what data is required and what information the module supplies. Even if the only reason is to split the design process within a team of modelers, they have to discuss the way the different parts will finally fit together.

Since an interface is a specification that standardizes the data exchange between the modules (or whatever the unit is referred to), all concerning entities have to conform to it. Depending on the kind of data exchange, bidirectional and unidirectional interfaces are distinguished. The former implies interdependence, which forces all modules precisely to know what the other side expects, whereas the latter requires a following module to react to the information provided by the sender.

For example, a band conveyor could be interchangeable with a bucket dredge. Both machines need to connect to an input and output device, but all process dependent code can be encapsulated in the module. The preceding module provides information without caring about the kind of transportation service. However, the conveyor as well as the dredge should let untouched the data they do not require. As the more general the interface gets designed the more different can be the functionality of the modules. It is good practice to provide more information than is actually required, since later modules could handle it.

In GPSS/H, the active, dynamic and universal simulation elements, called *Transactions* (*Xacts*), lay the foundation for the information exchange between the different modules and within the module itself. As the simulation time moves on, the Xacts move like the dynamic element they represent in the model. Thus, the interface has to provide a "bridge" for the transactions among the modules. Additionally, the interface specification has to contain the following components:

- The different kinds of Xacts handled by the module, which also includes the creation as well as the destruction of transactions. The first parameter of the data structure each Xact is carrying can be used to define and to identify the type. This approach requires each Xact to have at least this parameter.
- The data structure for each type of transaction modified or otherwise affected by the module.
- Whether and which module is allowed to modify a special date of the transaction created by the module. Sometimes the module is reliant on data that are created by itself and must remain unchanged.

If all modules provide this information, the simulation developer is able to perfectly arrange the model and to gain the information required without being precisely instructed what each module internally calculates.

In reality, the interfaces among the main modules of mine simulation example already mentioned consist of vehicles, such as trucks and tractors crossing roads, tunnels and bridges. The subsurface and surface area are physically linked by the road descending into and returning from the underground (see Figure 5.2). Each vehicle is represented by a transaction "loaded" with data, such as the kind and amount of material they are carrying, the amount of remaining petroleum and times when they passed several facilities. Thus, the interface actually consists of an invisible interruption in the road whereas the internal representation exists only as a formal specification of the transaction data structures.

As just outlined, the interface of a module becomes very important when it has to serve in different models and even different places within a model. As more information is supplied and as this information gets described better, more use can be drawn from the module. However, special attention should be payed when different modelers deal with the simulation.

Workgroup–orientated modeling

Although many simulations have been and will be written by only one modeler, large–scale projects often require more than one person working in order to be finished in a suitable period of time. The mine simulation project described in Chapter 5 has been developed by a small team of two, which, nonetheless, has showed the importance of a good coordination.

First, the project should be subdivided in parts as independent as possible in order to minimize the data exchange between them and thus to lessen the probability of mistakes which affect the other parts. As mentioned in the previous section, the interface has to be defined with care and foresight since even minor changes in the interface specification could cause major problems and efforts.

Referring to the practical example at the end of this paper, at the beginning of the project, the customer provided topographical maps of the whole mine including all facilities. These maps were a great help in understanding the complex relations and aided to develop an abstract plan to coordinate the modeling process. Because of the relatively small team, it was very easy to split this process. The underground and the surface area were connected only by a portal, an optimum condition to split the model at this point and to develop an interface. To further ensure the cooperation of these partial models, each important modification was discussed with the other team members to find the most efficient solution possible.

In large–scale simulation projects, however, it is usually necessary for more than one developer to be modifying modules at the same time. Bugs sometimes creep in when modules are modified and they are mostly detected a long time after the modification have been applied. It is actually quite easy to overwrite changes of the others unless one is extremely careful. Matters become even more worse when other modelers adopt the code.

Since these problems have been observed for quite some time, there exist some tools supporting the software developer (and in our case the simulation modeler) by keeping track of the changes team members did to the project. One of the more simple ones is *diff*. It locates changes in files of the project. Thus, it allows a developer to have a look at the modifications done since he had the code the last time. If something does not react the way it should, it might be caused by recent changes. The modeler, however, will be required to periodically save the project files or the differences between versions.

A much more sophisticated approach provides the *Concurrent Versions System* (*CVS* [Ber92]), which extends the notion of revision control from a collection of files in a single directory to a hierarchical collection of directories each containing revision controlled files. Directories and files in the CVS system can be combined together in many ways to form a software release. CVS provides the functions necessary to manage these software releases and to control the concurrent editing of source files among multiple developers. Every developer works in its own directory and CVS merges the work when each developer is done.

Such a tool cannot compensate for bad coordination and insufficient project arrangements. However, it makes the development process more secure.

2.2 Knowledge–based simulations

The model building process for a large–scale system usually requires a large amount of first–principle knowledge of the domain. It also requires heuristic and commonsense knowledge in order to determine the sets of phenomena to model as well as the appropriate procedure to achieve a given goal of the analysis. Since not all personnel participating in planning and maintaining the system have the same knowledge about it, each of them can derive benefits from software tools that are able to operate on a shared knowledge data base. The engineers, the simulation developer as well as the maintenance team, e.g., would provide their specific knowledge and simultaneously use the know–how of the others.

Figure 2.2 describes the principle architecture of a shared information system. The knowledge and data base do not have to be necessarily separated. For some aspects a common data base which stores information about the course of planning, characterization of the system as well as CAD geometry data would be more efficient. CAD and simulation applications would gain their data (in the form of knowledge–based rules, functional relations or simple object data) out of the same information base and thus support the data cohesion.

The knowledge base has to meet the following requirements:

- System–independent open architecture with standardized interface
- Possibility to store field–crossing knowledge
- Ability to manage access rights

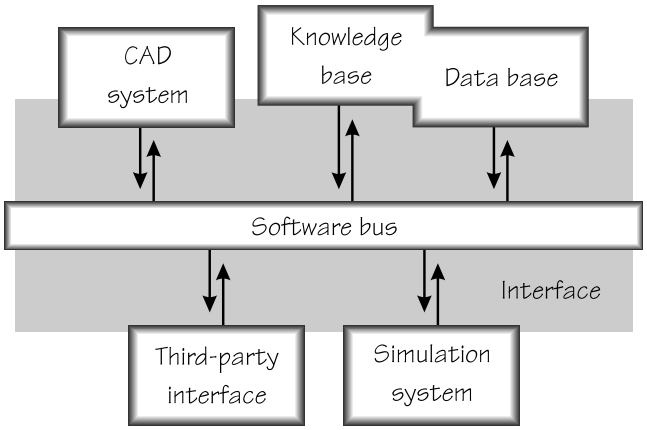


Figure 2.2: Knowledge–based simulation architecture

In spite of the different spheres of work, a lot of information is shared even if it is necessary to map or to adapt the required information. For example, the simulation developer may use equations describing the relations among parameters that govern the system behavior over time and apply them to the model. This information, however, could be supplied by the engineers who are already dealing with the various devices and environmental backgrounds. By standardizing of the information and the programs processing them, the software can operate on a single information base containing all data required to perform the project.

A serious approach to simulation is the *Joint Data Base Elements for Modeling and Simulation* (*JDBE* [JDB95]) project of the American Department of Defense, which has developed reverse engineering and data integration methodologies. The goal of the JDBE project is to promote the effective interoperation of models and simulations by means of data sharing achieved through information modeling and standardization. JDBE does this by developing candidate standard data elements and data models for subject areas in the M&S application field. By grouping data elements by subject area and applying a data integration methodology, it is possible to define the mappings required to share data among diverse data sources and user information systems.

The *STEP* standard for exchange of product model data, including the data description language *EXPRESS* [And90], has been introduced for CAD/CAM systems. STEP provides a standard data access interface to a neutral data exchange file and allows for the development of software which is able to exchange information with other applications in the process chain. EXPRESS serves as the definition of the product data model concerning STEP. It is based on an object orientated Entity–Relationship–Model[2] which is applicable to existing object ori-

[2]Standard model in the early planning stage of database development. For further information see [Heu95].

entated data models. A special feature is the possibility to define functions that operate on the complex data structures as well as integrity conditions for the objects. There exist various toolkits to handle EXPRESS models and to provide an easy to implement interface for several programming languages, such as the *ProStep* toolkit ([Pro97]) for instance.

This trend, also referred to as *Simultaneous Engineering* of production development and planning, is supposed to have consequences for the development of simulation tools. Standardized data interfaces and simulation structures, such as those provided by HLA (see also Section 1.3), may result in a new quality of simulation design and production engineering.

2.3 Distributed simulation

Distributed simulation is another approach to simulation which incorporates the ability to share large–scale simulations among geographically separated computers (simulators). The simulators are interconnected via a common communication architecture like the Internet and thus able to exchange simulation information.

There exist various vendor specific solutions. One approach, however, has become a standard as *Distributed Interactive Simulation* (*DIS*). Characteristically for DIS is its open architecture which allows systems developed by different vendors to operate within DIS as well as the modular design of its components. The latter also requires that each component connects to the overall system in a well defined and uniform manner. The benefits of a modular architecture have been already discussed in the previous sections. DIS also claims to be interactive. This aspect of DIS relates to the fact that within the simulated environment, people, machines and situational factors, to name only a few, can influence one another and vice versa.

Despite the fact that DIS was greatly influenced by the American Department of Defense (DoD) and is being developed for use within the military, there is a growing push to apply this technology to other settings such as factories, research laboratories and office environments. Probably, the most obvious use of DIS involves training. Providing individuals and groups the opportunity to perform activities and make decisions in simulated environments, can produce highly effective results. Another area where DIS can be applied is in the design and development process. By it, it will be possible to design virtual prototypes and to interact with them. The engineers, although geographically separated, could work together to ultimately reduce the cost and time normally necessary to develop a prototype.

It does not need much to imagine the potential DIS has for mining. The training of hazard situations such as accidents in the underground could save time and lives. Even in the development stage some aspects could be simulated and interactively experienced to discover problems. Virtual prototyping can help

to design optimized mine models before any cost intensive decisions are made. By a standardized communication structure and on base of a shared virtual prototype it would be possible to involve other person's knowledge into decisions.

Although the American army is using DIS for training in virtual reality environments, DIS is not primarily concerned with virtual simulations. Since its introduction, DIS has always been a dynamic system continuing to expand and incorporate new technologies. Thus it can be implemented in quite different forms of simulation environments. However, there are two documents, written by the DoD, which should guide the program development:

- The DIS Master Plan
- The DIS Modernization Plan

The recently introduced High Level Architecture (HLA) is intended to be the successor of DIS. This architecture provides some further approaches, such as to significantly reduce the network traffic caused by the provision of all possible information (broadcasting). Instead, each federate only supplies and receives the information that is requested. HLA has been designated as the standard technical architecture for all DoD simulations. However, in the transition, the common toolkits and simulation environments have to provide interfaces to both methods. There are also toolkits available which encapsulate the simulation and hide the actually used architecture from the model. Thus the same model can serve in both environments.

Chapter 3

Visualization of simulations

For quite some time now, simulators have been recognized tools to validate and to assist in decisions. Since the use of simulations supporting extensive projects is increasing, they need to be done in a way comprehensible for all who are participating in. Most simulations, however, lack appropriate presentation possibilities, which help to explain the processes and which facilitate the interpretation respectively the assessment of the results.

Usually, simulation results are presented in the form of tables and curves. Neither of these presentation forms are very attractive for the customer nor are they easy to assess, especially if the customer is not familiar with the reflected internals. Modern computers, however, are powerful enough to offer more attractive visualizations of simulation processes such as computer animations.

Animations aid in depicting simulation concepts and strategies visually and thus help to increase the acceptance of the simulation results. Furthermore, irregularities in the simulated system, such as supply shortfalls or deadlocks, are easier and quicker detected than by evaluating tables or curves. This chapter describes the different approaches and problems concerning the visualization of simulations.

3.1 Visualization by tables and diagrams

All of the broadly available simulators offer a way to present simulation results in the form of tables. This is the simplest method to obtain values out of the simulation run and, indeed, sufficient for small projects. However, large–scale models contain a multitude of information, which needs to be summarized and expressed by representations easier to access.

If the simulation system supports the output of data in a manner freely specified by the modeler, a third party application may perform the processing. Spreadsheet programs, e.g., are well suited to present and to further process numerical data. Most of them are able to import tables in the form of simple ASCII–files containing values tabulated by separators. Hence, the results of dif-

ferent simulation runs can be stored, compared and assessed with common office software. This might be the point the customer may put on his work. Macros developed to assist in evaluation and to facilitate repeating tasks can run on his computer. It enables the customer to process the simulation results by himself even if he is rather unfamiliar with the simulator.

Simulators with a graphical user interface also offer opportunities to represent the findings, which go beyond the aforementioned approach. Diagrams help to overlook a great amount of information and are a well suited aid to depict already assessed data. The conclusions of multiple simulation runs can be visualized in a way that the pros and cons for a specific modification can be seen at a glance. For instance, to compare the effects of a modification of one and the same value in different runs a set of pie charts may be displayed whereas bar charts better illustrate changes of a parameter within time periods. Graphs are more likely to be found in visualizations of continuous processes. However, discrete event simulation does not have to lack in this kind of representation (see also Chapter 4).

Since spreadsheet applications include methods to present data as diagrams in different ways, they emphasize their suitability once more. Despite the trend towards ever increasing complexity of the simulators leading to simulation systems able to analyze the simulation results as well as to present these, it will nevertheless be necessary and desirable to process the results with other planning and analyzing tools.

3.2 Different approaches for animations

Dependent on the level of abstraction, there are different possibilities to visualize the proceedings taking place during the simulation run. Often, a simple representation in the form of tables and diagrams is sufficient. Nonetheless, some processes need to be visualized in greater detail. Complex dependencies often appearing in large–scale systems are easier to observe in an animated representation of the processes.

With the availability of faster and more complex graphic systems the possibility to design more realistic representation arose as well, such as 3D process animations and Virtual Reality computer simulations. The latter technology is not yet broadly available. Nevertheless, realistic process visualizations are gaining in importance. For instance, the additional information transmitted by the third dimension may allow the observer to obtain supplementary conclusions in robotics (the origin of that technology) and in the field of factory planning and automation. On the other hand, 2D animations are better suited for the visualization of larger areas as they have to be modeled for traffic and transport simulations and where the space perception yields no significant improvements.

Aside the advantages, one should consider that the design of a three–dimensional

visual model is significantly more complex. Usually, mechanical engineers build their models with 3D design aids, whereas architects use 2D CAD software (if they use CAD at all). Simulation systems with 3D object–libraries are available, which help to accelerate the design process of the visual model.

However, even in the biggest library there will not be an object for each purpose nor will each representation fit the modeler's idea of what it should look like. Therefore, many animation tools have at least the capability of a simple drawing tool. Some of them contain a complete graphical construction program. Nevertheless, for large simulation projects it could be useful to adapt an existing CAD–layout. Using it, the simulation could be visualized in the same environment used by the project designer and architect.

Graphical simulation systems with object library

The simulation systems this section is concerned with differ significantly from other systems because of their ability to deal with the physical elements of a system in physical (graphical) terms and with the logical elements of a system in logical terms.

In the following, the discrete simulation system *Quest* (Deneb Robotics Inc.) is used as an example of a graphical simulation system with an object library. As with other systems of that kind[1], Quest provides a rich assortment of graphical representations for certain simulation elements, such as sources, buffers, machines and conveyors. The simulation model can be designed interactively by placing the elements on a construction plane. After connecting the relating elements, it remains to adjust the parameters of each entity to the appropriate values. In this way, the modeler builds a simulation model which automatically can be animated with no further effort. On the other hand, Quest also contains a powerful simulation language not only to implement special behaviors and to extend the functionality of certain objects but also to give the modeler full control over the simulation.

Figure 3.1 shows a screenshot of Quest simulating a simple production system with three flexible workcells capable of assembling three different kinds of products made of two components. The components Part_A, Part_B and Part_C are assembled with Part_D to form the end products Part_AD, Part_BD and Part_CD respectively. The workcells Mach_1, Mach_2 and Mach_3 require the appropriate components from two buffers to produce the end product on a cyclical basis. The assembled products are sent from the workcells by a common buffer after which they exit the system.

The example above is only a little beyond a minimal system and certainly not enough to assess the design procedure compared with classical simulation (see

[1]Choosing Quest does not constitute any rating. Simple++ (Aesop) and AutoMod (AutoSimulations), e.g., offer a similar functionality.

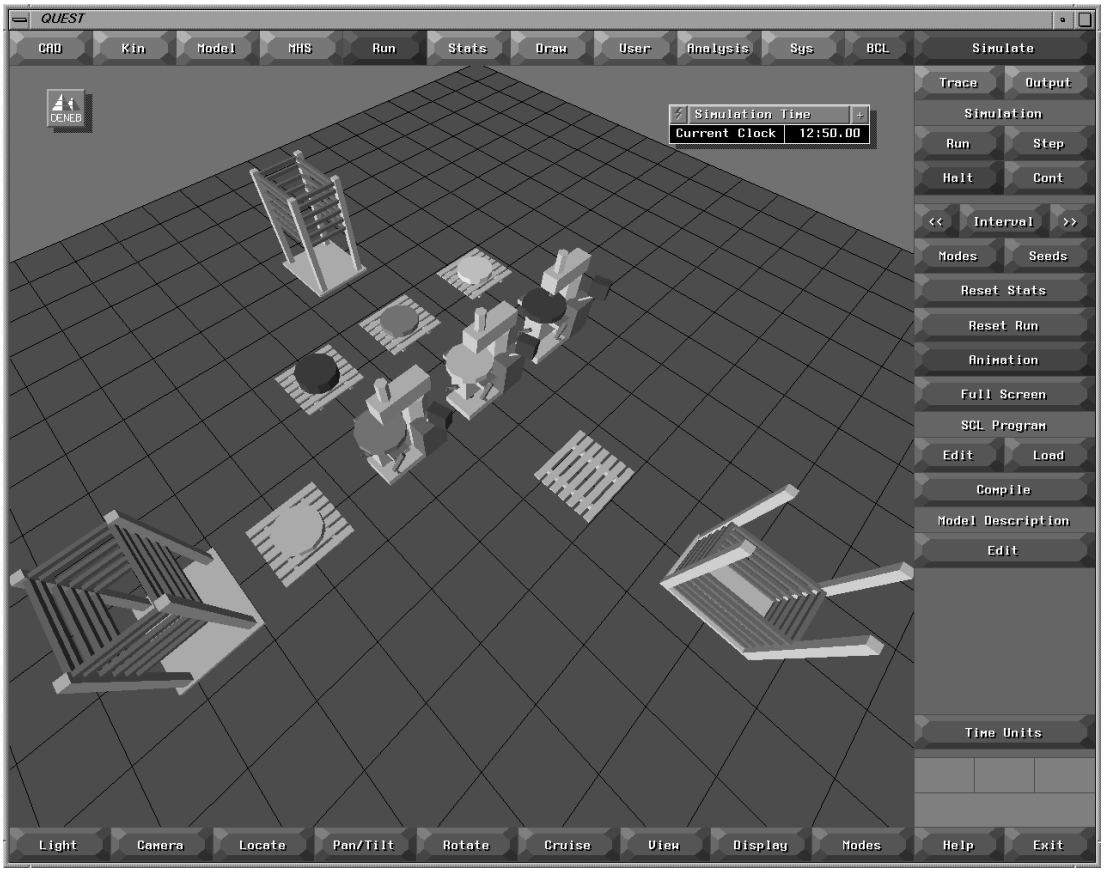


Figure 3.1: Screenshot of Quest simulating a simple production system

next section). However, it provides the possibility to show the different ways of simulation model creation.

To build such a model, the user chooses an element as needed from the appropriate menu and determines the graphical representation for the element. After that, the specific adjustments for each element, such as random distributions, delays or duration, have to be done. Finally, all related objects must be connected appropriately. In Quest, a source element is selected by the mouse and depending on its type an additional popup menu appears to give the user further choices. With the next selected object, Quest will set up a logical connection. When all adjustments are done, the system is ready to perform the simulation.

Quest also offers advanced features to allow users to manipulate the presentation. All graphics are represented in 3D space with unlimited viewing control including: translation, rotation, scale, light–sourced solids, perspective, and continuous motion viewing, whereupon the animation can be watched as being self a part of the animation. Furthermore, it does not lack in appropriate tools for analyzing the simulation results. Bar and pie charts extend the possibilities of visualizing the output values.

Special object representations for mining simulations can be imported from popular CAD–programs, such as *ProEngineer* or *AutoCAD*, whereas the built–in 3D drawing tool supports the creation of representations in Quest itself. Nevertheless, the need for an high end graphic workstation to perform a complex visual simulation shall be mentioned.

It is further worth considering, that large–scale models often exhibit special behaviors that cannot be treated by canned objects. Consequently, the modeler has to fall back on the integrated simulation language in order to implement special cases. Obviously, the simulation system loses a lot of its advantages. However, for recurrent tasks, object library based simulation systems are a very effective way to write visual simulations, although they often have disadvantages concerning the runtime[2]. Since most of them allow one to design custom objects, the invested effort may be used in the future again.

Classical simulation systems with graphical extensions

In contrast to simulation systems with an object library, the classical simulation systems provide their services in the form of a programming language. Following, *GPSS/H*[3] is taken as a representative for this category which is also the simulator used for the practical example of Chapter 5. GPSS/H itself offers no direct opportunity to animate simulation proceedings nor does it provide other graphical representations of simulation results. It has to rely on graphical tools that can

[2]Because of the multipurpose concept, the objects contain handlers for a multitude of events, which not only improve the flexibility but also affect the performance of the simulation.

[3]General Purpose Simulation System in a special implementation by J.O. Henriksen. For further information see [HC].

be controlled by prerecorded commands. Since most of the common presentation programs already contain a macro language, this restriction sometimes turns out to be an advantage in flexibility.

As previously mentioned, a simulation language gives the modeler full control over the whole simulation, but it also requires him to know and to consider the internal proceedings of the simulator, which often have no direct connection to the simulation project. Therefore, many object library simulators try to keep the modeler away from the internals, whereby he is free to focus on the simulation itself. However, large–scale models often contain complicated logic, which can be controlled more effectively by a simulation language. Moreover, the programmer has the opportunity to construct an efficient and optimized logic leading to significant advantages concerning the runtime of the simulation.

To give an example, Listing 3.1, printed at the end of this section, shows the GPSS/H simulation code of the production system described in the previous section (see also Figure 3.1). For better clarity, the model has been lengthened. The three sections simulating the machines only differ in labels. Thus, it would be relatively easy to combine them in a macro. At the end of the listing, the simulation presents some final results via the `BPUTPIC` statement. In this special case, the modeler wants to analyze the maximum length of all buffers as well as the average use of the machines.

```
104           BPUTPIC    FILE=RESULTS,LINES=7,(QM(BUFFERA),QM(BUFFERB),_
105 QM(BUFFERC),QM(BUFFERD),FR(MACH1)/10,FR(MACH2)/10,FR(MACH3)/10)
106  max length BUFFERA **
107  max length BUFFERB **
108  max length BUFFERC **
109  max length BUFFERD **
110  average use MACH1 *.**%
111  average use MACH2 *.**%
112  average use MACH3 *.**%
```

Instead of presenting the values on the screen, they also could be tabulated and stored in files for further processing using a spreadsheet or the modeler can record the activities of the objects by using commands for an animator such as the one mentioned next.

Proof–Animation (Wolverine Corp.), hereinafter also referred to as *Proof*, is a 2D graphical animation tool containing features such as path animations, collision detection and multiple views. It is mentioned here because of its ability to be driven by commands spooled in a file called *trace–file*. The layout that has been set up separately includes objects representing the active components of the model. These objects, e.g., can be moved and modified in shape by the external commands. Since GPSS/H can output arbitrary data, it also may record proceedings occurring during the simulation. However, the modeler has to care about all creations, movements and destructions of animation elements which can lead to considerable extra work depending on the extent of the animation. This drawback is the biggest disadvantage of this kind of system, namely, the

absence of an interprocess communication. By it, the animation program could share information about collisions, distances between moving objects and states of elements with the simulation system. On the other hand, the visualization gets significantly improved in speed as the simulation does not need to run during the presentation. Consequently, the program performing the visualization is completely independent from the simulator which makes it easy to use different computers not at least in view of a presentation to a customer. Moreover, the customer does not get static data, such as a set of transparencies. Rather he can freely focus on the aspects he is interested in without being familiar with the simulator itself.

The dependencies among the objects of a large–scale models are often hard to observe for the modeler. However, minor inaccuracies can turn into major ones concerning the questions the model should help to answer. Thus, an animation can also be an useful tool to validate and to debug the simulation.

There are two ways to create the layout for Proof–Animation. The first refers to the paragraph above and consists in the ability of Proof to process commands. Since the layout file is built of instructions representing the vector graphics the layout consists of, GPSS/H can record these as well. However, this will be the exception. The other way is to use Proof's integrated drawing tools which is sufficient for smaller animations or ones with little detail. More complex representations of objects can be assembled from simpler drawings and stored as a class. Therefore, it is possible to build a graphical object library. Furthermore, classes are necessary to use paths for guiding. Proof itself controls the movement of objects (represented by classes) on the paths according to the speed each object got assigned. While the simulation does not need to update the location of the moving objects, it has to care about what goes next. Hence, the visualization of the proceedings can easily amount to multiple work than necessary for the simulation as such.

Proof–Animation is also well suited to visualize a simulation in a CAD–created environment. The integrated DXF import filter supports the conversion into Proof–owned format. Especially if a CAD layout of a mine or a factory building exists, it is quite vivid to show the simulation in the same layout that the architects and engineers use. This leads to a coherent planning environment with less need for explanations.

The simple but powerful simulation language GPSS/H makes the implementation a straight forward task. However, it does not provide any object–oriented approaches which force or help the modeler to design a structural model. Instead the modeler is expected to meet some guidelines. Since the simulation and animation are completely independent, changes in the simulation code often require extensive work in the animation layout and vice versa. Therefore, the code needs to be structured otherwise changes may cause major problems.

Above all, graphical extensions have the ability to visualize simulation proceedings and the results of simulation systems that do not offer graphical repre-

sentations. Many of the older simulators provide no or only limited possibilities to produce animations directly but are still powerful enough to be used ongoing.

Listing 3.1: GPSS/H simulation of a simple production system

```
 1 * GPSS/H simulation of a production system, which assembles three different
 2 * kinds of end products on three available flexible workcells
 3
 4           SIMULATE
 5
 6           REALLOCATE COM,25000             // allocate additional memory
 7
 8  RESULTS  FILEDEF    'RESULTS.DAT'         // define results file
 9
10  FALSE    SYN        0                     // synonyms for the values of
11  TRUE     SYN        1                     // a boolean variable
12
13  PARTA    SYN        0                     // synonyms for the different
14  PARTB    SYN        1                     // components
15  PARTC    SYN        2
16  PARTD    SYN        3
17
18  RSTREAM1 SYN        1                     // synonyms for the independent
19  RSTREAM2 SYN        2                     // streams of random distributions
20  RSTREAM3 SYN        3
21  RSTREAM4 SYN        4
22  RSTREAM5 SYN        5
23  RSTREAM6 SYN        6
24  RSTREAM7 SYN        7
25
26  MACH1RDY BVARIABLE  ((CH(BUFFERD)'GE'1)AND(FS(MACH1)))  // boolean variables
27  MACH2RDY BVARIABLE  ((CH(BUFFERD)'GE'1)AND(FS(MACH2)))  // to test the state
28  MACH3RDY BVARIABLE  ((CH(BUFFERD)'GE'1)AND(FS(MACH3)))  // of the machines
29
30  ASMTIME  FUNCTION   PB1,E3                // function to compute the
31 PARTA,RVEXPO(RSTREAM5,40)/_                // different durations of the
32 PARTB,RVEXPO(RSTREAM6,40)/_                // assembling processes
33 PARTC,RVEXPO(RSTREAM7,40)
34
35           INTEGER    &XACTCNT,_                    // var to assign components
36                      &BUFCYL                       // var to distribute -"-
37
38           LET        &XACTCNT=RN(RSTREAM3)@3       // initalize the variables
39           LET        &BUFCYL=RN(RSTREAM4)@3        // with random values
40
41           GENERATE   RVEXPO(RSTREAM2,15),,,,,1PB   // SOURCE_D
42
43           ASSIGN     1,PARTD,PB                    // assign the sort
44           QUEUE      BUFFERD                       // like a buffer in Quest
45           LINK       BUFFERD,FIFO                  // put the parts in a chain
46
47           GENERATE   RVEXPO(RSTREAM1,15),,,,,1PB   // SOURCE_A,B,C
48
49           ASSIGN     1,&XACTCNT@3,PB               // assign the sort
50                                                    // @3 == mod 3
51           BLET       &XACTCNT=&XACTCNT+1
52           BLET       &BUFCYL=&BUFCYL+1
53
54           TEST  L    &BUFCYL@3,2,BUFFERC           // distribute the different
55           TEST  L    &BUFCYL@3,1,BUFFERB           // parts onto the 3 buffers
56
```

```
 57  BUFFERA  QUEUE      BUFFERA                        // buffer the part
 58           TEST  E    BV(MACH1RDY),TRUE              // is the machine ready ?
 59           SEIZE      MACH1                          // occupy the machine
 60           DEPART     BUFFERA                        // leave the buffer
 61
 62           UNLINK     BUFFERD,*+2,1                  // get one PART_D
 63           TRANSFER   ,DOASM1                        // go to the asm process
 64           DEPART     BUFFERD                        // PART_D leaves the buffer
 65           TERMINATE                                 // we've no further
 66                                                     // interest for PART_D
 67  DOASM1   ADVANCE    FN(ASMTIME)                    // compute the asm time
 68           RELEASE    MACH1                          // release the machine
 69
 70           TERMINATE                                 // the assembled part
 71                                                     // could be used further
 72  BUFFERB  QUEUE      BUFFERB
 73           TEST  E    BV(MACH2RDY),TRUE
 74           SEIZE      MACH2
 75           DEPART     BUFFERB
 76
 77           UNLINK     BUFFERD,*+2,1
 78           TRANSFER   ,DOASM2
 79           DEPART     BUFFERD
 80           TERMINATE
 81
 82  DOASM2   ADVANCE    FN(ASMTIME)
 83           RELEASE    MACH2
 84
 85           TERMINATE
 86
 87  BUFFERC  QUEUE      BUFFERC
 88           TEST  E    BV(MACH3RDY),TRUE
 89           SEIZE      MACH3
 90           DEPART     BUFFERC
 91
 92           UNLINK     BUFFERD,*+2,1
 93           TRANSFER   ,DOASM3
 94           DEPART     BUFFERD
 95           TERMINATE
 96
 97  DOASM3   ADVANCE    FN(ASMTIME)
 98           RELEASE    MACH3
 99
100           TERMINATE
101
102           GENERATE   ,,,1                  // the defined duration for the
103           ADVANCE    1*3600                // simulation run (1h)
104           BPUTPIC    FILE=RESULTS,LINES=7,(QM(BUFFERA),QM(BUFFERB),_
105 QM(BUFFERC),QM(BUFFERD),FR(MACH1)/10,FR(MACH2)/10,FR(MACH3)/10)
106  max length BUFFERA **
107  max length BUFFERB **
108  max length BUFFERC **
109  max length BUFFERD **
110  average use MACH1 *.**%
111  average use MACH2 *.**%
112  average use MACH3 *.**%
113           TERMINATE  1
114
115           START      1
116           END
```

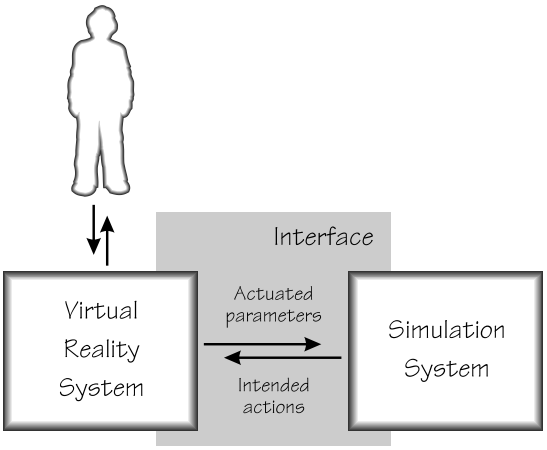


Figure 3.2: Combination of Virtual Reality and simulation

Virtual Reality and interactive simulation systems

Interactive simulation systems, which enable the user to interact with the running simulation, have a high potential for further improvements of simulation models. Virtual Reality[4] (VR) as a technology to actively involve someone into a virtual model can serve as the interface between the users and the simulator. The observer is not limited to only watching the proceedings in the simulation similar to being in a cinema. Instead he or she may be part of it. He or she can modify locations of elements or change other parameters during the simulation process to react to results he or she already got in this run. Complex and lengthy simulations can be shortened as the user does not need to restart the whole process because of small modifications. In fact, he or she may optimize the layout of a plant without interrupting the simulation while the simulator adapts the simulation to the new settings.

As illustrated in Figure 3.2, the Virtual Reality system takes care of the user's interaction with the model. The results of interactions, such as the actuated positions of virtual objects, are transmitted via the interface to the simulator. The changes in the model will immediately take effect and the simulator will send updated versions of the intended actions to the Virtual Reality system.

Indeed, that kind of simulation system does much more than what many simulation projects actually seem to need. If, for instance, one only wants to analyze the traffic volume in the underground tunnels of a subsurface mine, a Virtual Reality visualization of all activities there would certainly be overdone. However, dangerous or high realistic situations can be simulated and studied with the factor human. Virtual prototyping, in mining as well, leads to shorter develop-

[4] In [Bau96] Virtual Reality has been defined as follows (translation): Virtual Reality is a technology allowing real–time interactivity within a three–dimensional computer model. By techniques of immersion, the user gets the subjective sentiment of being in a real environment.

ment periods which makes it easier to react to the discovery of new deposits for instance.

Many Virtual Reality simulators exist as distributed application to facilitate the implementation of the simulation and visualization tasks. Furthermore, it gives the opportunity to independently update each part and to distribute the application among different computers. Particularly the Virtual Reality environment highly depends on the hardware performance of the host machine. However, before a user may interact with the simulation the model has to be build. The complete three–dimensional environment, which includes all objects the user may interact with as well as the information about the relations between them, needs to be modeled. For this purpose, special Virtual Reality modeling programs are available, which help to design polygonal optimized models and also offer special functionality related to Virtual Reality.

If a CAD layout of the model already exists, some work might be saved by using it. However, even if the CAD drawing is three–dimensional, it is seldom made for a Virtual Reality system and often contains too much detail. But the level of detail is crucial for the performance of the visual simulation. As more polygons are used in the scene as more computing power is needed to avoid jerky motions during interactions. The goal is to retain the appearance of the objects as natural as possible in spite of reducing the amount of polygons significantly. *Clarus CAD Real–Time Link* (Prosolvia Clarus AB), a tool designed for this purpose, automatically removes and joins polygons in regions specified by the user to simplify the model but also to save as much detail as possible. Generally, in Virtual Reality rough but textured surfaces are much better suited than fine molded polygon surfaces. Textures convey very realistic impressions which are hard to model.

MultiGen Pro (MultiGen Inc.) is a very sophisticated tool to design a Virtual Reality environment. This application supports the modeler in the whole process starting with the creation of a simple wireframe model up to the applying of textures or the subdividing onto different levels of detail.

Up to now, there are no broadly available Virtual Reality simulation systems. The American army is one of the motors of this technology (see also Section 1.3 and 2.3) and for a long time has been the only serious user. However, other companies are doing research to apply the advantages of Virtual Reality to non-military applications. Figure 3.3 shows a Virtual Reality simulation system being developed at the Fraunhofer IFF. *SOS–VR* (Self–Organization Simulation[5]) simulates production facilities in a Virtual Reality environment and allows interactive interventions. In the last few years, much has changed on the market for Virtual Reality software. More and more toolkits are appearing which provide an extensive functionality which keep the developers implementation on a high level.

[5]In the self–organization simulation each object is described by the capability to create attraction fields and by the sensitivity for these attraction fields. For further information see [FVU97] and [VFS+97].

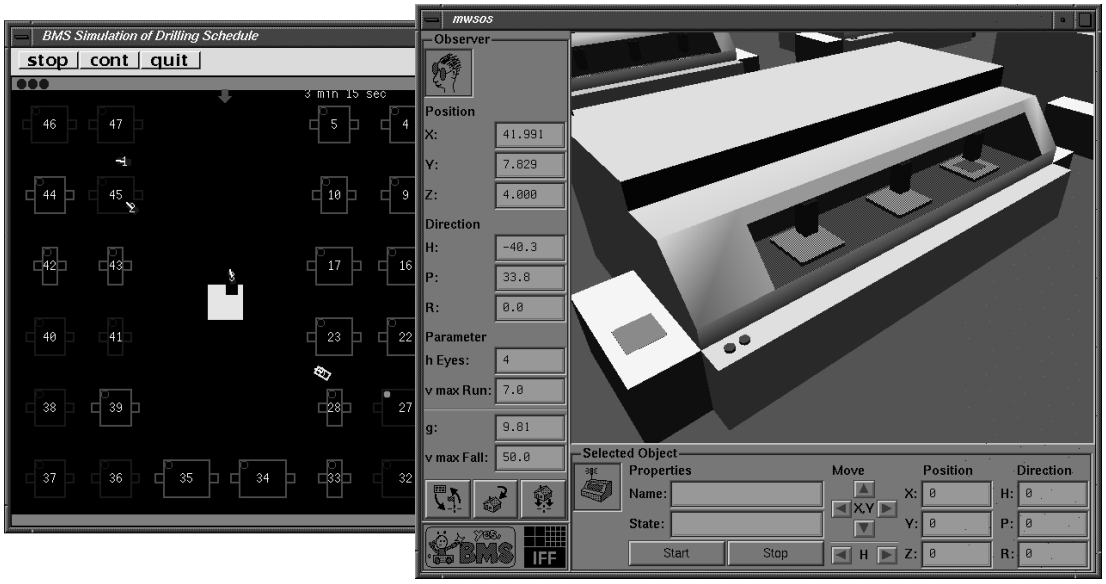


Figure 3.3: SOS–VR — a Virtual Reality simulation system, courtesy of Fraunhofer IFF

Vega–VR (Paradigm Simulation Inc.) and *SmartScene* (MultiGen Inc.), to name only two of them, provide an API to extend their functionality which is already able to perform a Virtual Reality presentation without any lines of code.

In the future, Virtual Reality simulation systems will probably gain in importance, since the concept provides a new way to combine simulations with the possibility of real–time user interactions and thus to allow the analysis of required personal participation in the simulated process. Furthermore, the visualization also helps to explain the proceedings to people not familiar with abstract simulations.

3.3 Adaptation of existing CAD layouts

As seen in the previous sections, an existing CAD model could be very helpful in the design of the visualization of a simulation. However, depending on the available data and on the kind of representation chosen, some postediting is required.

Most of the drawings will be 2D ones produced by the project architects or factory engineers. Thus, in most cases, the CAD layout will be designed for the construction rather than for the layout of a simulation. For construction plans it is insignificant that due to the number of layers used during the design process there are a lot of overlapping lines. However, for the animation, these overlapping lines can cause problems. To build paths out of the drawing is much more complicated

if there are duplicated lines. For instance, the animation tool is not able to decide which of the lines it should take to construct paths. Animation systems allowing movable cameras also have to deal with these unnecessary graphic primitives slowing down the whole presentation process.

It is good practice to eliminate layers with redundant drawings and to join similar layers. Proof–Animation, for example, produces many classes grouping arbitrary layout elements if the source layout contains different layers. These classes are then included in the animation layout which impedes straight forward changes. Furthermore, if the CAD design contains more colors than the animator selected can provide, some color deviations will occur. To prevent the animation system from redefining colors, these adjustments should be done in the CAD program. As mentioned in the previous section, there also exist some special tools to reduce the complexity of a given layout. However, lower complexity designs can also remain unchanged. As shown in Chapter 5, Proof–Animation is an excellent visualization system for existing 2D CAD designs. Within the imported layout, the simulation designer can place paths to connect facilities and machines in order to animate transportation systems or to show other logistics.

An existing 2D CAD layout could also simplify the design process of a 3D model. Most of the 3D design aids and animators are able to import popular 2D formats. After removal of inscriptions and the like, the design can serve as a master or template. The sketch may be assigned to a base plane. According to the side elevation, the ground area can be extruded and so converted into a 3D model. In general, for complex models an already existing CAD design can significantly reduce the time necessary to create the model.

3.4 Layout optimization and its visualization

Occasionally, an aim of a simulation project is to find the best arrangement of facilities, such as the best place for a machine or the optimum transport route between stations. Since the objects do not exist independently, such a problem cannot simply be solved in one step.

On principle, the object has to be moved, all connections and interdependencies updated and the simulation and animation must be performed again. However, this procedure is hardly applicable to complex problems as they arise from large-scale models, since it would be too time consuming and quite complicated due to the dependencies.

Mathematical optimization methods meet this iterative procedure by changing model parameters to obtain the extreme value of a target function. An integration of these methods into the simulation system allows the automatic calculation of the best arrangement. This problem has been studied in numerous articles with the result that analytical methods from differential and variational calculus

are not as suitable as numerical methods such as the *Powell* or *Hooke-Jeves*[6] algorithm.

[Kuh93], describes optimizations concerning 3D simulation models of assembly cells. These very flexible cells can accommodate robots and tables to assemble power switches, pneumatic valves or gearboxes for drilling machines. The robots, which are capable of preparing and assembling parts, have to be situated to provide the shortest production cycle considering in addition, the minimum number needed is sought. This is a very complex problem, since the program controlling each robot has also to be altered during the simulation in order to perform the correct assembly steps.

To visualize the optimization process and to make the steps more transparent, it would be useful to go beyond the basic animation of the assembling process for a certain configuration. Rather, an animation of the modifications occurring during the layout optimization process may be helpful to find the optimum configuration, which does not necessarily mean the best. Usually, there are also other influences to consider and not all of them can be utilized in the simulation.

According to the demands of such visualization, it might be sufficient to draw only a graph of the modifications, showing the movements concerning the last position. However, by recording the layout changes for each element in full detail or by bounding boxes, further conclusions could be drawn. After this procedure has been applied, the optimized layout will be available as well.

Another approach consists of using an interface to a mathematical optimization process. This can be a part of the simulation environment itself or an independent mathematical engine. The High Level Architecture (see also Section 1.3) provides, with its Runtime Infrastructure, excellent conditions to combine an HLA–written simulation with such an external math engine.

Instead of just calculating the optimum arrangement, an interactive graphical simulation may help involve the human within the simulation. Thus, it may be possible to represent influences which are hard to model. In state of the art Virtual Reality simulation systems, the user can interact with the simulation during its execution. For instance, the user arranges the components and modifies the layout while he or she observes the effects of these changes. It is also not too difficult to imagine that a real worker equipped with a data–glove and a head–mounted display could be involved in the simulation. The worker would perform his tasks as usual and simultaneously control the simulation in a way as to effect the real system. In fact, he or she may also discover problems relating to circumstances, which have been not deliberately modeled or which other persons would not consider. It is no secret that the employee is the one who knows best about the positive or negative influences on his work. (see also Section 9)

[6] Direct search method (numerical optimization)

Chapter 4

Continuous approaches in discrete simulation systems

Within discrete simulation mechanisms there are a number of logic expressions that are evaluated at discrete points in time. There is another approach to simulation known as *continuous*. With continuous simulation, time is controlled by continuous variables expressed as differential equations.

Depending on the kind of system to be simulated and on the questions it should help to answer, some components could probably be more easily represented in a continuous simulation system than by discrete mechanism. In an ore mill, for example, there are event–based proceedings, such as the start and stop of the milling as well as breakdowns and scheduled maintenance, whereas the continuous ore flow is expressed by differential equations (e.g. $dx/dt = 59\ [t/h]$). Although some simulation systems basically enable the user to mix continuous and discrete modeling, there are still some advantages of a pure discrete approach not at least with regard to an element–oriented animation.

This chapter will show how continuous simulation approaches can be reproduced within a discrete simulation system.

4.1 Different approaches to modeling

In reality, time and space are both represented by continuous functions. Since discrete event simulation can only approximate continuous behavior, it might not always fit best. The representation that will ultimately be chosen may also depend on the specific tasks to analyze. For example, the tunnel system of a subsurface mine may be modeled either as a conveyance system providing transport services or as a passive facility requiring all transport occurrences to be handled separately (Figure 4.1). If greater emphasis is given to continuous processes, such as the transport capacity utilization or the amount of ore mined per hour, representations by differential equations might be better suited, whereas discrete evaluation simplifies statements about single entities and their correlating events.

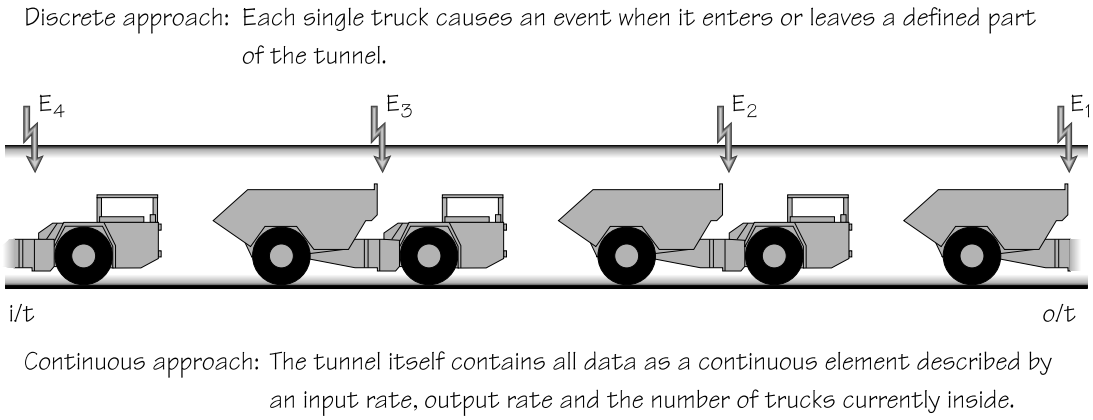


Figure 4.1: Example of a discrete and continuous simulation approach

Large–scale systems, such as plants, are usually too complex to develop an exact mathematical model of all components. Furthermore, sequential controls and the intervention of human operators cannot simply be mathematically integrated. Hence, it will be necessary to combine both approaches.

[Aka95], goes beyond and proposes sequential control as an additional type of model, beside continuous process and discrete event. According to his thesis, it is more convenient to handle sequential control separately, since it has properties of both discrete event and continuous process simulation.

4.2 Adaptation of continuous processes

The adaptation of continuous processes to discrete simulation systems raises the problem of reproducing the differential equations by single events. Continuous as such means determinable at any point in time. The effect within the discrete simulation system, however, is the division of time into single events, between which nothing exist. Since these are the only times computation has been done for, nothing can be said about the time in between. Due to practical limits, only a finite number of "legal" times can be evaluated and thus queried to draw conclusions.

In calculus, a continuous function can be approximated using a number of very small steps. As these steps are reduced in size and increased in number, the approximation becomes closer and closer to the true function. In the limit of an infinite number of steps of zero size, the approximation becomes an exact match for the function ($\int f(t)\,dt$).

Except for the infinite and zero amounts, this is what has to be done to adapt continuous processes to discrete simulation systems. Referring to Figure 4.2, the time axis is associated with the simulated time variable t of a process–dependent

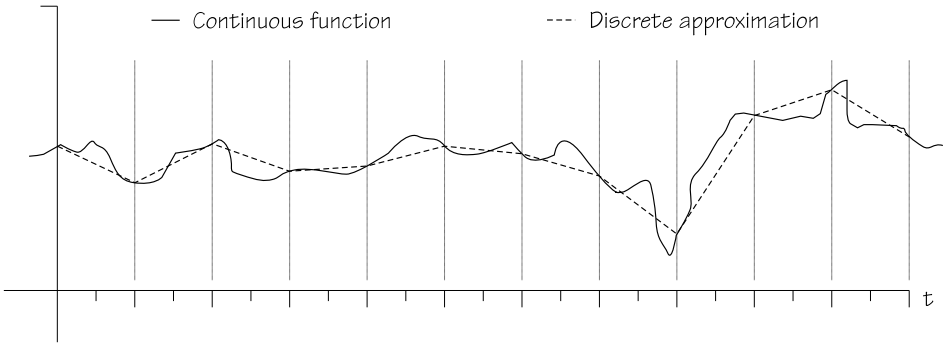


Figure 4.2: Discrete approximation of a continuous function

function. Every step of the simulation adds a small, but not infinitesimal, amount dt to the time. Thus, the total time in the simulated world is equal to the sum of all dt's. The only difficulty is to find the right time increment, which is a tradeoff between accuracy (small time increments) and speed (large time increments).

To demonstrate the transformation on a practical example, the already considered mine example from Chapter 5 shall be taken once more. The ore mill, although a continuous system, has been integrated to the discrete simulation model. For a more detailed description of the mill and the surroundings see Section 5.2.

Abstractly seen, the mill does nothing else than continuously splitting the incoming crude ore into its ingredients at a specific input and output rate. Conveyors and devices which transport, crush, press or otherwise affect the ore flow can also be modeled by differential equations. In fact, the parameters of such devices are mostly expressed by those equations. The present millrate of 59 tons per hour can be reproduced by 59 discrete events per hour, each of them representing 1 ton of ore. When the millrate is changing, the delay for the next event increases or decreases as well. Since the size of the unit is crucial for the exactness of the results, it is essential to consider the amount and frequency of variations in the specific parameters. The higher they are the shorter must the delays be to ensure precise analysis. For the mill, 1 ton units have proven small enough.

4.3 Visualization of continuous processes

The simulation results of continuous processes are usually depicted by curves in the course of time. As discrete simulation only approximates continuous functions, the results actually cannot be represented by curves. Strictly seen, it is not allowed to connect values if the related parameters are not continuous. However, as seen above, the differential equations of continuous processes can be adapted to discrete simulation systems and thus visualized as shown.

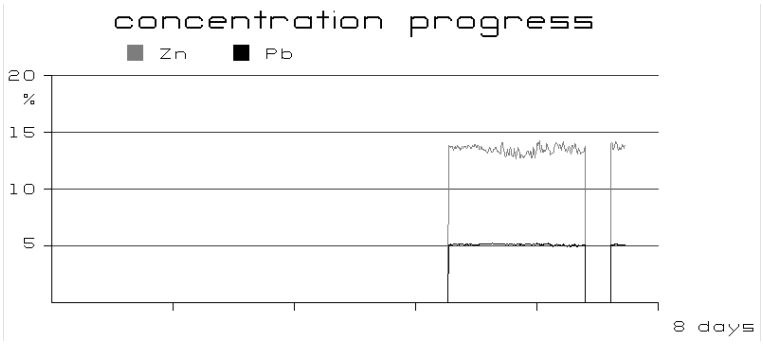


Figure 4.3: Continuous simulation results expressed by curves

If the intervals between the values are relatively short in view of the whole time axis, predictions about the values in between will meet most requirements. One should consider the possibility of the visualization tool to zoom into a curve which then should result in a more detailed view of the curve.

In the early stages, the aforementioned mill simulation also contained a plot of continuously changing variables (Figure 4.3). The curve depicted the concentration of zinc and lead within the ore after leaving the ball–mill. Here, the interval between two successive values was short enough to connect them by a line. However, depended on the underlying expression, other representations, such as polynomials of higher degree, may be better suited to interpolate the intermediate values.

Chapter 5

Mining simulation with GPSS/H and Proof–Animation

The mining simulation described next has been mentioned several times in the previous chapters, and is due to a project performed for the Kennecott Corporation Utah/USA. *Kennecott's Greens Creek Mine* is a lead and zinc ore mine located on an island near Juneau, Alaska. It is currently undergoing a major expansion project.

To provide a model that could assist in decisions and more clearly describe the proceedings and processes, the mine had to be simulated and visualized in all important aspects. Thus, the mining, the hauling, the milling and the shipping of the concentrate are all shown together on same animation. This has provided an interesting example for a general mining simulation and also raised some interesting problems. Some of these problems and approaches shall be mentioned in this chapter.

5.1 Subsurface area

The main facility, "the heart" of each mine, is the place where the ore comes from. Since most of the other activities in the mine are highly dependent on it, the validation of decisions due to any modifications are highly important. A simulation can help to improve the reliability of modifications by visualizing the effects.

Motivation

In this mine, which can be regarded as a typical subsurface mine, the mining process is quite difficult due to complicated logistics in the tunnels. All of the tunnels are too narrow to allow two vehicles to pass. To lessen the effect of this bottleneck, which could obviously be the reason for a delay, there are some bays, hereinafter referred to as *Passing Bays*, along the tunnels. In addition to

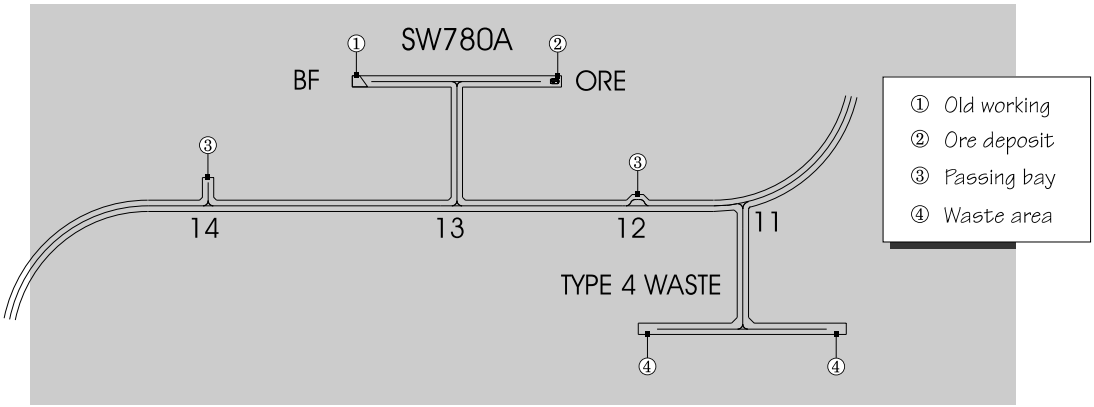


Figure 5.1: Section enlargement of the subsurface area showing the main components

these passing bays, the trucks can also use the various turn outs into the mining areas but doing this would block the mining areas. Figure 5.1 shows the main components of the subsurface area as they appear in the animation.

To guarantee the supply of ore to the processing facilities and to reduce the number of vehicles in the tunnels, the returning trucks always have the priority over these coming from the surface. The latter can also use a special tunnel, the *Loop Road*, to drive faster to more far away mining areas. The simulation should demonstrate whether or not this loop would significantly relieve the congestion in the main tunnels.

Because of the narrow tunnels, an accident or breakdown would partially paralyze the underground traffic. The simulation had also to show the effect of all these possible situations. The discovering of supply shortfalls and bottlenecks, especially in the case of underground breakdowns, was one of the main aims of the simulation.

Furthermore, there were considerations to use ore trucks for backfill transports in addition to the backfill trucks. Although it seems to be a good idea to use them too, the relations are fairly complicated. For instance, if there would already be enough backfill trucks working on this task, the ore trucks have to wait too long at the backfill plant. On the other hand, it would increase the efficiency of the system to use less trucks in the mine in view of the underground traffic. In this connection, the capacity of the different trucks matters as well. The model allows to adjust both to the users need.

At the time when the model was developed, the mining took place in two shifts from Monday through Saturday. However, it was planed to further increase the output of the mine by introducing a third shift as well as mining on the Sundays also. Hence, the model also reacts on changes of the shift system.

Some mining areas are shut down or are used to store waste. To analyze the

complex reactions due to changes in the tunnel system coupled to shutdowns of certain areas, the simulation needs to handle this also.

Last but not least, the simulation should show the possibilities of mining simulation and animation for the mine, and furthermore help to visualize the complex relations and processes in an easy, comprehensible way. According to comments by experts in mine simulation, this has been the first fully simulated and animated underground mine.

General structure

Figure 5.2 shows a rough sketch of the architecture of the subsurface area as appearing in the animation. The lines in the middle of the tunnels, normally hidden, represent the paths that guide the different vehicles. The intersection at the very top of the illustration connects the subsurface area with the mining facilities, namely, the mill to the left and the waste areas straight ahead. Starting from this intersection, all paths and tunnels are too narrow to allow two vehicles to pass. As mentioned before, vehicles coming from the underground always have priority. Hence, the tunnel system has to provide other passing opportunities. However, the narrow tunnels do not only affect the descending vehicles. A truck traveling up is further obstructed as the truck ahead of it causes delays. The faster truck has to slow down and to follow in line.

All vehicles, descending or returning ones[1], have to cross Greens Creek by a small bridge next to the intersection. Between the bridge and the portal, before the vehicles descend into the underground, the first of the 38 passing bays is located — the only one on the surface. There are two types of passing bays, one which allows laterally passing whereas a truck entering the other needs to drive backward like into a parking space (Figure 5.3). Additionally, the entrances of the mining areas as well as the entrance to the maintenance bay may be used for passing also. To sum up, all branch tunnels can be occupied by returning vehicles.

To prevent collisions and to secure smooth traffic, each descending vehicle has to check whether or not it will be the only one in the next tunnel section, whether or not a vehicle is coming in front of it and whether or not the next passing bay is unoccupied. Only if these conditions are all true, may it enter the next section of the tunnel. In the same way, it has to check the path if it wants to drive out of a passing bay. This ensures a clear way for all returning vehicles and speeds up emergency operations in case of an accident or a breakdown.

The loop road is exceptional due to its one–way character. Descending vehicles may use this loop to drive faster to more far away located mining areas. Since this tunnel exists on paper only, the tunnel may be closed or opened to traffic.

[1]The term *descending vehicle* is used in the following to express that this vehicle comes from the surface whereas a *returning vehicle* comes from the underground and heads for the surface.

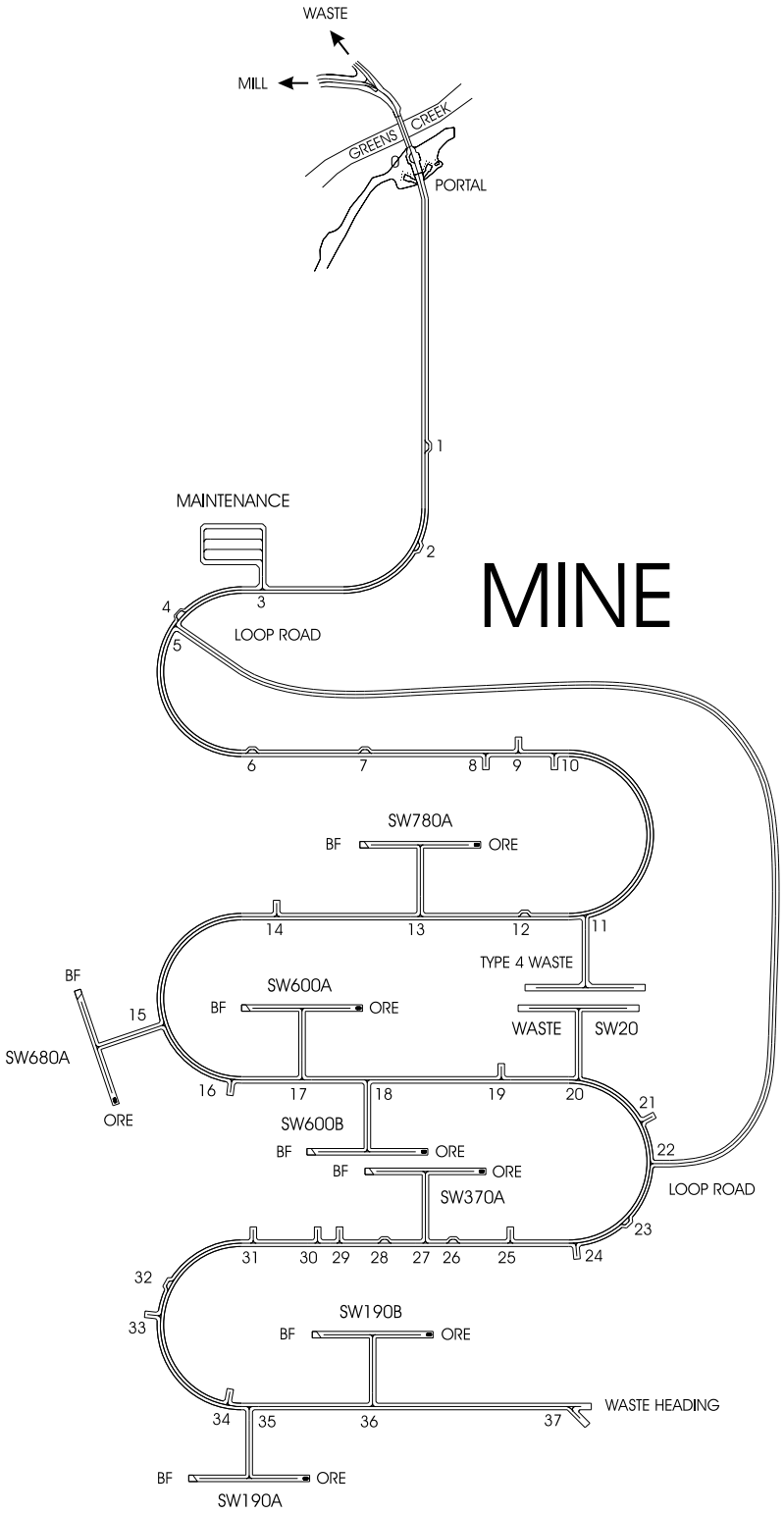


Figure 5.2: Layout detail showing the subsurface area of the mine

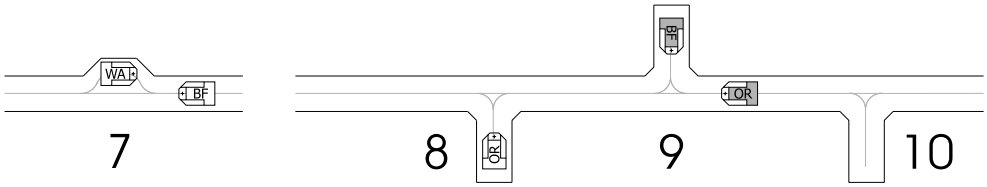


Figure 5.3: The different types of passing bays

During shift changes, all vehicles in the underground area have to drive into the *Maintenance Bay*. The different kinds of vehicles line up at three parking areas, one for tractors, one for backfill trucks and one for ore and waste trucks together. As the name of this bay implies, the trucks and tractors are checked and refueled. Vehicles requiring further maintenance can be repaired as well. The ore and waste trucks stay the all day Sunday in the bay. When the next shift begins, all teams commence the work at this point.

The areas where the ore actually gets mined, referred to as *Mining Area*, branch out to both sides of the main tunnel. To depict the dynamic changes that result from the mining and loading processes there, a dump sign stands for the area that has been shut down and which needs to be filled whereas the digger sign refers to the area that is still being mined (Figure 5.1).

The old mining areas are filled with backfill material, which is a mixture of tailings from the ore mill and cement produced at the backfill plant near the mill. The backfill trucks load the mixture, bring it to the appropriate mining areas and dump it. On the way back they have nothing to transport.

The crude ore is carried by the ore and empty waste trucks which transport it to a stockpile near the mill. Usually, the ore trucks descend without any load into the underground but, as mentioned earlier, there were some considerations to use the ore trucks for loading backfill too. Although it would lead to a better utilization of the ore trucks, since they could bring backfill from the surface to the underground and drive up with ore, there are also other aspects to consider. The simulation, however, allows one to investigate both approaches.

Three areas in the underground differ from the others. Two former and now closed mining areas (*Type 4 Waste* and *Waste SW20*) as well as the area at the end of the main tunnel marked *Waste Heading* are used to store waste. Waste material is brought from the surface to these special areas by waste trucks. Unlike the other trucks that often drive without any loads, the waste trucks pick up ore on their way back. Only if there are already trucks in all mining areas between the waste area and the portal, do the waste truck returns without any loads.

Tractors work on all the supplementary duties, such as bringing workers and the material into the underground.

There is no special selection plan or order by which the trucks choose their destination. Ore, backfill and waste trucks drive to the first free section.

In order to be able to analyze the behavior of the mine with regard to the production and time as well as the traffic volume, the number of the different kinds of vehicles can be varied. Furthermore, the whole subsurface area has been designed to investigate exceptional cases such as breakdowns and accidents and their relating effects. The connections among all the facilities are numerous, which make it quite difficult to make decisions without simulation.

Simulation details

The layout shown in the animation is based on the original construction plans, which have been developed using AutoCAD to design the mine. To facilitate the import of CAD data into Proof–Animation, the layer structure as well as unnecessary information were removed (see also Section 3.3). The original layout, for instance, contained some geological descriptions, which were not as important as the information that had to be added to support the animation. Aside from providing space for objects appearing during the animation, the design has been simplified to facilitate the perception of the complex processes.

To guide the moving objects, such as trucks and tractors, Proof–Animation requires the modeler to define paths in the layout. As mentioned, this simulation project was thought to be the first fully simulated and animated mine model for a complete miming and milling operation. In order to validate this claim, the animation had to emphasize the detailed representation of all the various components of the mine and mill system. Hence, vehicles drive around smooth curves instead of jumping around corners, events, such as parking, loading, unloading or the transport of workers, have been animated to name only a few. About 300 paths had to be defined in the underground area alone.

To give a further example of the efforts spent on the visualization of the whole mining operation in the underground area (the surface model has been developed with the same effort but is subject of another paper[2]), the operating cycle of an ore truck that transports backfill as well as ore is described in the following (see Figure A.1 for assistance).

The ore truck starts a shift at the backfill plant. If more than one truck wants to load backfill material, the others have to line up. After receiving the mixture of cement and ore tailings, the ore truck drives to the intersection and stops there until the way in front is clear. In order to come to the subsurface area, it has to turn left. At the junction next to the Greens Creek bridge, the truck needs to make sure that no other vehicle is on the bridge and that the passing bay behind it is unused too. Only if these conditions are fulfilled, it may descend into the underground. Every time an ore truck enters a new tunnel section, it must check if another vehicle is coming up. If so, it must move into the next bay. If the loop road is open to traffic, the truck may use this short cut to drive to the lower

[2] This paper is split into two reports. For further information about this project, such as the surface and the dock area, see [Göt98].

mining areas unless a vehicle is blocking the entrance. After all, the ore truck turns around in the first unoccupied mining area, where it dumps its load. The shovel in the ore section also requires the truck to reverse in order to load ore. The truck needs to turn again within the middle section. Once it is loaded, it leaves the site and heads for the surface but has to check first if a vehicle is using the gateway as a passing bay. Since returning vehicles have priority, it only has to stop if a descending one is already in the section it wants to enter or if a vehicle broke down in front. After the bridge, it crosses the junction and dumps at a stockpile near the ore mill. If gas is low, it drives to the gas station. The cycle is closed by returning to the backfill plant. If, however, the ore truck has to queue, it may also return without backfill to the underground area.

All vehicles in the tunnels move at random speeds according to their purpose. To prevent returning vehicles from running into or jumping over each other as well as from colliding with descending ones, quite complicated logic had to be used. Since Proof–Animation performs the visualization based on data recorded during the simulation, all possibilities had to be catered for the simulation itself (see also Section 3.2). Hence, vehicles which bottle up behind a slower one must adapt to the speed by logic instead of by being assigned a new speed when the animator discovers collisions. To handle these events, the following logic was devised, which *only* deals with vehicles returning to the surface.

- Each tunnel segment (altogether 39) contains a counter to keep track of the number of moving vehicles within its section. Additionally, the first returning vehicle entering the segment sets a flag for the descending ones indicating that this section is already used and, furthermore, assigns its speed value to an internal path variable. The last truck or tractor which leaves the segment removes the flag.

- Each returning vehicle entering a segment checks whether another vehicle is already on the path. If there is, it may need to wait in front of the next passing bay until the descending vehicle has moved into the bay. If the vehicle already on the path is also ascending, it may need to adapt to the speed of the one ahead. If, for whatever reason, a vehicle is blocking the next section, it stops until the way is free again.

Listing 5.1 shows the expanded macro of the GPSS/H logic just outlined. The next paragraph describes the simulation code for the section between passing bay 13 and 12 (see Figure 5.2).

Before reaching this code fragment, the simulated vehicle leaves the previous segment starting at passing bay 14 and is ready to pass mining area SW780A. The first `QUEUE` was added in order to obtain statistical information about the volume of traffic as well as to determine the number of trucks currently waiting in front of this section. The next `GATE` only allows a truck to pass if no truck is on the path ahead. Otherwise, the truck must check that the one in front has gone

some set distance (TEST in line 3). The following double GATE prevents the vehicle from driving into a disabled truck as well as into an oncoming one. After storing the time when the next truck may enter the section, the sequence on lines 11–13 cancels the registration of the current vehicle in the previous tunnel section. If it has also been the last one there, the flag indicating the presence of an object (logic switch AA141) has to be cleared as well. The code on lines 15–18 sets or resets a temporary flag depending on whether another vehicle is in this section. The car finally enters the segment and passes bay 13. The BPUTPIC instruction writes the next three lines into a file defined by ATFFACE, which records the proceedings for Proof–Animation. PL5 is a variable tied to the simulated object (see Section 5) and stores the current speed of the vehicle. The label BACK24 allows a truck to enter the path after the bay when an ore truck turns out of the mining area SW780A. SWTKBKDN is a macro concerned with trucks that have broken down. The GATE below stops all vehicles in this tunnel segment if one has broken down. As soon as the truck is repaired, it continues and updates the time when the next one may enter the section. The GATE in line 37 refers to a flag that was set if a vehicle has already entered the segment and not yet left. if so, the truck adapts to the speed of the one ahead. Otherwise, it is assigned a new speed value according to its type (lines 41–44). Last but not least, the control commands for Proof–Animation are written and the car checks whether another truck is waiting in front of the next tunnel section.

Listing 5.1: Code fragment controlling a tunnel section for returning vehicles

```
 1  BA24     QUEUE       TQ13TO12         // register waiting trucks
 2           GATE LS     AA131,*+2        // someone on the path in front ?
 3           TEST GE     AC1,&LUB13B12    // enough space for the truck ahead ?
 4           GATE LR     NOUP1312         // a broken truck on the path ?
 5           GATE LR     AA122            // someone on the path in the
 6           TRANSFER    SIM,,*-2         // opposite direction ?
 7           DEPART      TQ13TO12
 8
 9           BLET        &LUB13B12=AC1+&FOLLOWUP  // time for the next entry
10
11           BLET        &N1413=&N1413-1          // leave the previous path
12           TEST E      &N1413,0,*+2             // was it the last truck ?
13           LOGIC R     AA141                    // clear the presence indicator
14
15           GATE LS     AA131,*+3                // set a flag if there already
16           LOGIC S     AA131USD                 // is a car on the track
17           TRANSFER    ,*+2
18           LOGIC R     AA131USD
19
20           BLET        &N1312=&N1312+1          // enter the next path
21           TEST E      &N1312,1,*+2             // was it the first one ?
22           LOGIC S     AA131
23
24 * Proof-Animation: set truck on path           // pass the passing bay
25           BPUTPIC FILE=ATFFACE,LINES=3,(AC1,XID1,XID1,PL5*&PASSBA13)
26  TIME *.****
27  PLACE * ON UpPassBA13
28  SET * TRAVEL *.****
```

```
29            ADVANCE     PL5*&PASSBA13            // consider the duration
30
31  BACK24    ADVANCE     0                        // jump on the path
32  SWTKBKDN  MACRO       PH10,PH3,SEGM:13-12,NOUP1312  // handle breakdowns
33            GATE LR     NOUP1312                 // hold the broken truck
34            TEST E      PH54,1,*+2               // did thisone fail ?
35            BLET        &LUB13B12=AC1+&FOLLOWUP  // time for the next entry
36
37            GATE LS     AA131USD,*+3             // is there a car in front ?
38            BLET        PL5=&B13TOB12            // adapt to its speed
39            TRANSFER    ,A1312
40
41            TEST NE     PH10,WASTETRK,B1312      // get random speed
42            BLET        PL5=RVTRI(203,&SWE1312A*&MIN,&SWE1312A,&SWE1312A*&MAX)
43            TRANSFER    ,A1312
44  B1312     BLET        PL5=RVTRI(204,&SWE1312B*&MIN,&SWE1312B,&SWE1312B*&MAX)
45
46  A1312     ADVANCE     0
47            BLET        &B13TOB12=PL5            // store the speed of the truck
48
49 * Proof-Animation: set truck on path
50            BPUTPIC FILE=ATFFACE,LINES=3,(AC1,XID1,XID1,PL5-(PL5*&PASSBA12))
51  TIME *.****
52  PLACE * ON BA13toBA12
53  SET * TRAVEL *.****
54            ADVANCE     PL5-(PL5*&PASSBA12)
55
56            TEST E      Q$TQ12TO11,0             // someone waiting ahead ?
```

By restricting the name length to maximum 8 characters, GPSS/H forces the modeler to devise a quite complicated system for the names of variables, logic switches, queues etc. As just outlined in the code fragment above, for a person not actively involved in the development process, it is difficult to understand the mnemonics used.

Figure 5.4 shows an abstract plan, which has been drawn to keep control over the multitude of logic switches and facilities used to reproduce the traffic system in the underground. Although the information on the plan seems to be recurring and a short algorithm recorded on a much smaller sheet could probably replace the whole sketch, there are differences. However, most important has been the chance to obtain an overall view of the traffic system. Furthermore, the sketch helped to prepare and to coordinate the modeling process of the whole subsurface area.

The left column describes the tunnel starting from the portal to the end whereas the other column illustrates the opposite way (see also Figure 5.2). To aid in explaining the elements, a section enlargement is shown at the right. The number `5` may represent a passing bay, the turn out of a mining area or, as in this case, the gateway to another tunnel. The number `52` stands for the logic switch `AA52` used to signal that a descending vehicle has moved into segment `SEG0506` (the section from bay `5` to bay `6`). Accordingly, the number `51` represents the logic switch `AA51` indicating that a moving element has entered segment `SEG0504`. The terms `BA32` and `BACK32` are labels, which are destinations to move to within a tunnel. The former leads to a position directly in front of the gateway to the

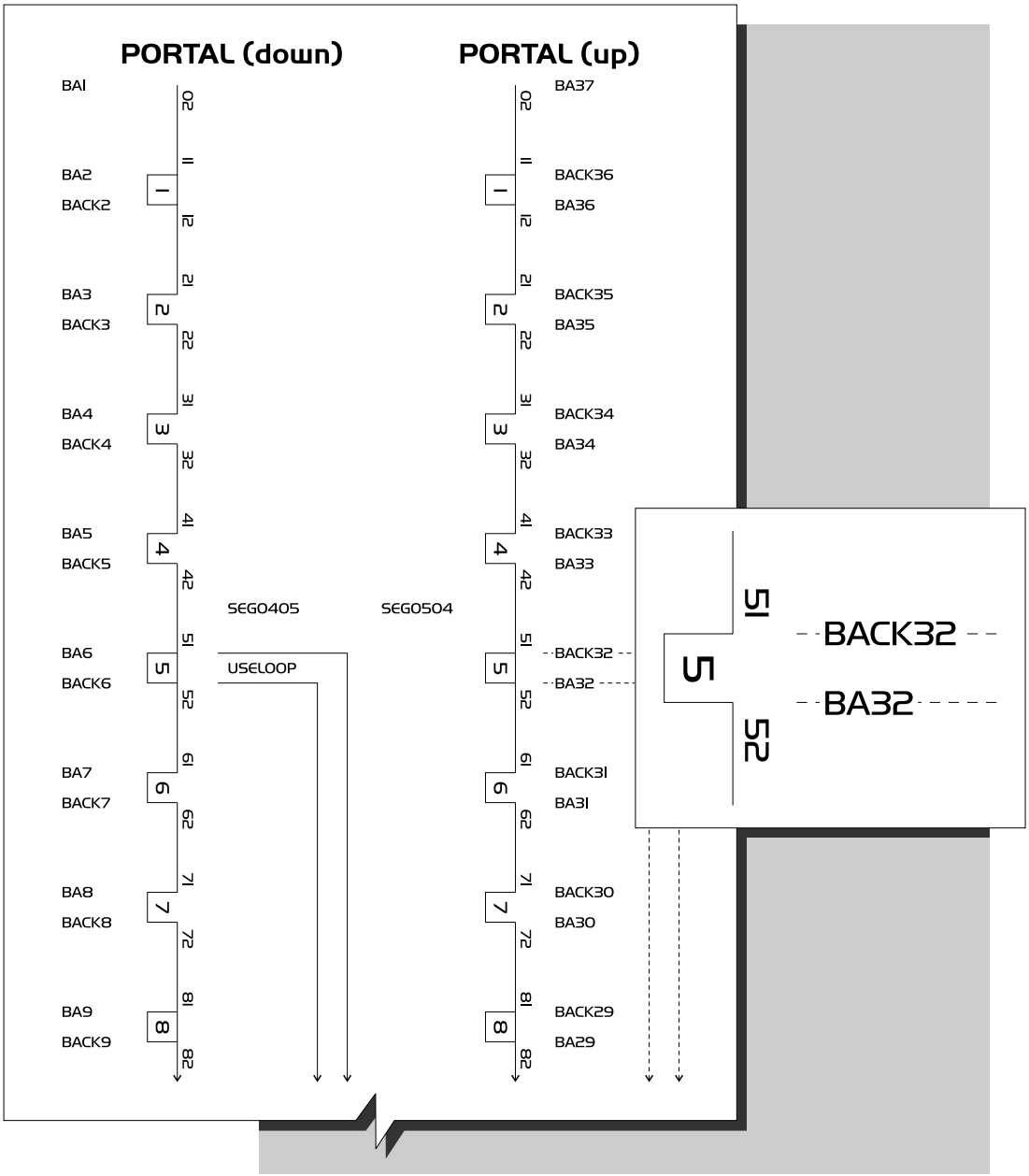


Figure 5.4: Abstract plan of the tunnel system regarding the traffic control logic

loop road whereas the latter allows a truck to join the path exactly behind the gateway. The five elements just described are by far not the only ones controlling the movements in a tunnel segment. Rather they are the ones used and tested in other parts of the model. Beyond it, some names of variables are composed of these numbers. In summary, it can be said that such a plan or sketch is very useful and pays for itself.

In order to facilitate the modification of the simulation parameters, the control variables have been separated to files. Hence, it is possible to precalculate some values by other methods and to keep the simulation data and model separate. Furthermore, the customer can easily modify the behavior of the model to react to changes without being familiar with the simulation system itself.

5.2 Ore mill

The surface operations also involve a mill that processes the crude ore into concentrate, which then can be further processed. Tailings from this mill are trucked back to the mine to be used as backfill. The concentrate is taken by trucks to the dock area where ships will haul it to the smelter. As the second most important and complex facility, the mill is also one of the plant components where high emphasis has been placed. Thus, the mill has been modeled and animated in detail.

Motivation

Aside from just being animated, the model should also help to investigate the mill with regard to dimensioning and sizing of its components. For instance, the storage tanks have to be big enough to cushion the effect of a breakdown of the related presses down–stream. This means that they should not cause the shut down of the mill due to an impending overflow. Thus, the model offers parameters allowing to control and to change the behavior of the mill concerning these aspects.

Despite all shifts and downtimes, the mill has been designed to run continually. The simulation should prove this and, if necessary, help to find solutions to avoid bottlenecks. Obviously, the other components that affect the mill also had to be considered within this context.

Although just being a means to an end, the integration of a continuous subsystem, which the mill is an example of, into a discrete model has been challenging (see also Chapter 4).

General structure

Figure 5.5 provides an structural view of the mill components and how they are connected.

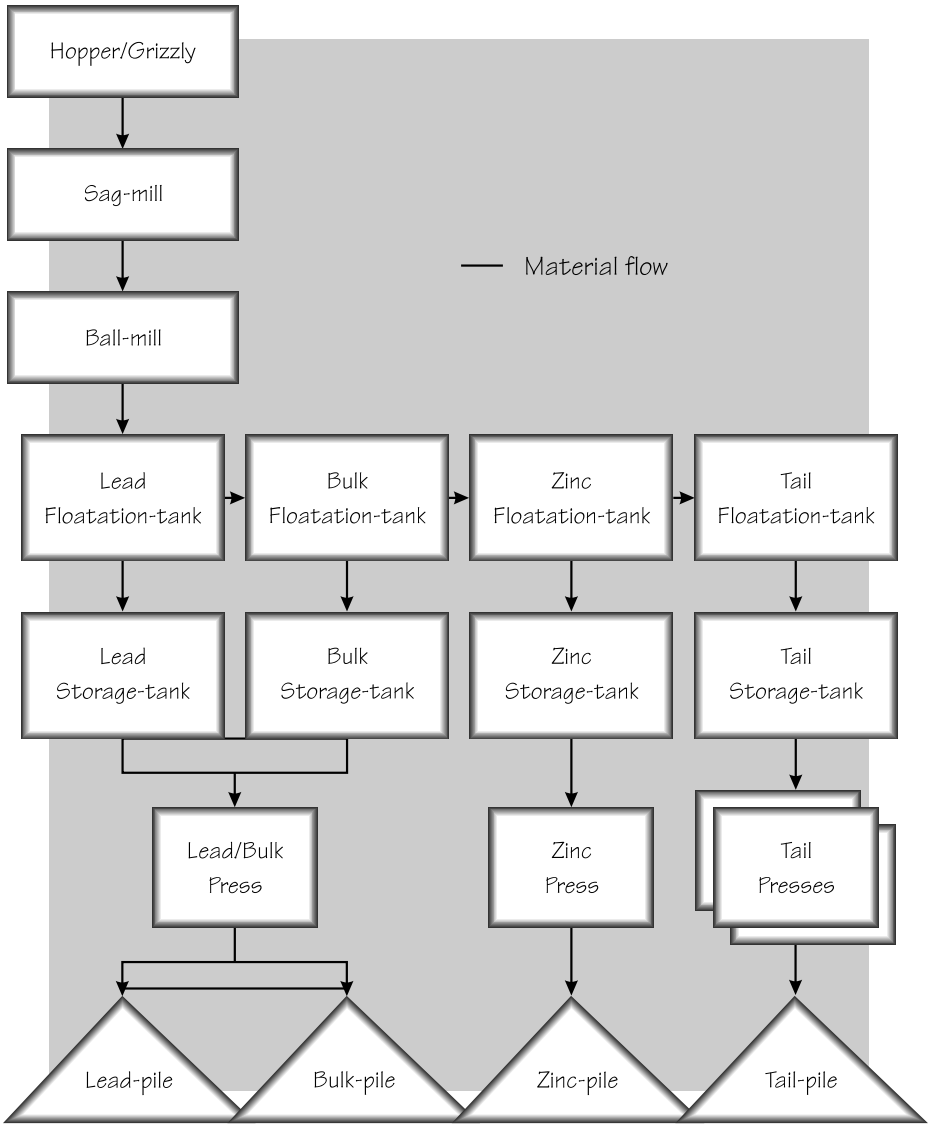


Figure 5.5: Flowchart of the milling process

To supply the mill with raw material, the ore trucks, and, under special conditions the waste trucks as well, are loaded with crude ore in the underground and return with it to a stockpile near the mill. This stockpile consist of a fixed number of pads (currently 2) with a specific capacity. A front end loader transports the ore to the hopper as soon as the hopper level sinks under some lower threshold. The crusher also starts at a specific hopper level. The crushed ore is then conveyed to the sag–mill[3] where the ore is milled until the percentage of lead and zinc in the mixture is within set limits. The ore moves from the sag–mill to the ball–mill[4] which crushes the ore mixture once more. Both mills are controlled by a feedback loop which occasionally speeds up or slows down the milling process in order to achieve a tolerance. Via the floatation tanks the out–coming mixture is divided into four different ingredients (lead, zinc, bulk and tail) and, accordingly, split into storage tanks. To be able to ship the raw material later, the contents of the tanks have to be dried by three presses. These presses need periodically maintenance while the mills keep running. Obviously, the storage tanks also serve as buffer for the presses. If, for whatever circumstances, the volume in one of the tanks reaches the maximum level, the milling process is slowed down as it would happen when the hopper goes empty. If the level is exceeded, the entire mill is shut down. Lead and bulk are dried and pressed at the same press, to better use its capacity. This happens alternately or accordingly to the level in the tanks. The resulting concentrates are dumped onto separate piles.

The main attention is directed to the continuous processing of the crude ore. The conception of the mill components has to guarantee the uninterrupted run of the physical and chemical processes.

Simulation details

Most of the values representing the current state of the components are expressed over time. The sag–mill, e.g., is not only implemented by a static capacity but also by a dynamic millrate specified in tons per hour. These continuous parameters are of great interest since their composition embodies the overall input and output rate of the mill. However, *continuous* means determinable at each time. As this mine model has been written in GPSS/H, a discrete simulation language, the continuous equations had to be reproduced by single events in a way which allows one to analyze the behavior without significant distortions. The interval between two successive events is the smallest indivisible unit conclusions can be drawn from. Because nothing can actually be said about the time in between,

[3] A *sag–mill* is a large cylindrical mill which has nothing in it but the ore itself. The ore is rotated and smashes as it drops to the sides. The pieces that come out of the mill are less than 1 inch in size.

[4] A *ball–mill* is a cylindrical mill which uses high strength molybdenum balls to further reduce the size of the pieces. The ore is rotated until it becomes quite fine — down to the size of small sand grains.

this interval has been carefully determined. Compared with the amounts of ore flowing through the devices per hour, the time a ton of ore is handled by them has proven to be small enough to ensure precise statements.

Beside scanning the values in the right interval, precise analysis requires precise modeling. Since the ore taken from the pads and loaded into the hopper comes from different mining areas, the ratio of lead and zinc varies as well. Hence, the composition of the ore conveyed to the sag–mill, e.g., does rarely correspond to the amounts of the ingredients coming out of the ball–mill. The variations in quality, even if they are not amounting to large values, have a certain impact on the proceedings, such as the millrate. Considering these aspects, the GPSS/H model of the sag–mill below supplies detailed information about the composition inside.

Listing 5.2: GPSS/H model of the sag–mill

```
1 *  sag–mill subroutine
2 *
3 *  PL1 – percentage of lead (Pb) in the ore
4 *  PL2 – percentage of zinc (Zn) in the ore
5
6  SAGMILL1 BLET       &OBSS=&OBSS+1                 // bump observation ++
7           TEST L     &OBSCNTS,&SAGCAP,CKWRAPS      // if vector isn't yet full,
8           BLET       &OBSCNTS=&OBSS                // update bump count
9  CKWRAPS  TEST G     &OBSS,&SAGCAP,NOWRAPS         // if obs # is > 25,
10          BLET       &OBSS=1                       // wrap around to 1
11
12 NOWRAPS  BLET       &RUNS1=&RUNS1–&VECTS1(&OBSS) // back out old value
13          BLET       &RUNS1=&RUNS1+PL1             // add in new value
14          BLET       &VECTS1(&OBSS)=PL1            // replace old value
15          BLET       &RUNAVS1=&RUNS1/&OBSCNTS      // calculate present average
16
17          BLET       &RUNS2=&RUNS2–&VECTS2(&OBSS) // do the same for zinc
18          BLET       &RUNS2=&RUNS2+PL2
19          BLET       &VECTS2(&OBSS)=PL2
20          BLET       &RUNAVS2=&RUNS2/&OBSCNTS
21
22          TEST E     &OBSCNTS,&SAGCAP              // wait until vector is full
23          TRANSFER   ,PH(2)+1                      // return
```

Basically, the ore inside the sag–mill can be considered as split into columns each containing 1 ton. Hence, the statistics associated with the material coming out of this mill correspond to the smallest unit available and thus are used for control of the logic. Each time a ton comes into the sag–mill or leaves the sag–mill, the average amount of the ingredients inside changes.

The millrate is controlled by several factors. As mentioned above, the ratio of the ingredients is one of them. If, for instance, the percentage of zinc leaving the ball–mill exceeds some lower or upper limit, the millrate, which also determines the feedrate, gets adjusted. Another factor is the fill level of the tanks. If it reaches 80% of the maximum, the millrate is successively reduced by 20% until the level becomes normal. Afterwards, the millrate is increased by the same amount to get higher output.

Since this simulation was mainly done to forecast the plant behavior in exceptional cases, scheduled maintenance and unscheduled breakdowns play an important role in the model. Some of the components need periodically maintenance every 12th, 4th or every week. The mill is also shut down for set times. Much harder to plan, however, are the sudden breakdowns. As mentioned above, the presses concentrating the different ingredients are crucial for the tanks upstream. Thus a breakdown will soon paralyze the whole mill process. Two of them are shut down for maintenance daily but an average of about 800 hours a year the presses are out of work due to unscheduled breakdowns. The fragment below lists the segment controlling the down times.

Listing 5.3: Segment causing scheduled and unscheduled downtimes of the presses

```
1           PRESSES    FUNCTION RN400,C2         // presses are numbered 41–45
2           0,41/1,45
3
4           DAYMINS    FUNCTION RN390,C2         // return random time of day
5           0,0/1,1440
6
7  ****************************************************************
8  *  segment to shut down 2 presses daily for scheduled maintenance
9  ****************************************************************
10
11          GENERATE   &DAILY,,0,,20,1PH,3PL
12          SPLIT      1,*+1                     // create two XIDs
13
14          BLET       PH1=FN(PRESSES)           // press to service
15          BLET       PL1=FN(DAYMINS)           // time starting maintenance
16          BLET       PL3=RVUNI(402,1.2*60,10)  // time required to fix
17
18          ADVANCE    PL1                       // hold Xact till it's time
19
20          BLET       XL(CUMSCH)=XL(CUMSCH)+PL3 // sum of scheduled downtimes
21
22          BPUTPIC    FILE=PRESS,LINES=1,SYM(FAC,PH1),AC1/60.,PL3/60.,_
23 &DAYTYPE(&OWDAYNO),XL(CUMSCH)/60.
24  *******,****.*,HR,PRESS DOWN,**.*,HRS,***,DAY,SCHEDULED  ,CUML=,****.*,HRS
25          BCLOSE     PRESS
26
27          FUNAVAIL   PH1,CO             // let current process finish
28
29          ADVANCE    PL3                // wait until press is fixed
30
31          FAVAIL     PH1                // make press available again
32
33          TERMINATE
34
35 ****************************************************************
36 *  segment to shut down presses due to unscheduled breakdowns
37 ****************************************************************
38
39 *  &MUDNHRYR – mill unscheduled downtime [hours/year]
40 *  &DAYSNYR  – days in year (usually 365)
41
42          GENERATE   ,,,100,,2PH,2PL
43
```

```
44           BLET       PH1=FN(PRESSES)                        // press to service
45           BLET       PL1=(FIX((FRN404)*&SIMDAYS)+1)*24*60  // starting time
46           BLET       PL2=RVTRI(57,10,90,12*60)              // time to be down
47
48           BLET       &TOTUNSCH=&TOTUNSCH+PL2   // sum of unscheduled downtimes
49
50           TEST LE    &TOTUNSCH,(&MUDNHRYR/&DAYSNYR)*&SIMDAYS*60,NOTNEDED
51
52           ADVANCE    PL1                       // hold Xact till it's time
53
54           BLET       XL(CUMUNSCH)=XL(CUMUNSCH)+PL2 // sum of unscheduled dntms
55
56           BPUTPIC    FILE=PRESS,LINES=1,SYM(FAC,PH1),AC1/60.,PL2/60.,_
57 &DAYTYPE(&OWDAYNO),XL(CUMUNSCH)/60.
58  *******,****.*,HR,PRESS DOWN,**.*,HRS,***,DAY,UNSCHEDULED,CUML=,****.*,HRS
59           BCLOSE     PRESS
60
61           FUNAVAIL   PH1,CO            // let current process finish
62
63           ADVANCE    PL2               // wait until press is fixed
64
65           FAVAIL     PH1               // make press available again
66
67  NOTNEDED TERMINATE
```

Beside the mill–internal down times, the breakdowns of the supply or the concentration trucks are important. However, the concentrate piles, e.g., have a much higher capacity than the floatation and storage tanks and impacts, such as those resulting from a press breakdown, need not be considered. Since the ore mill is only one of the mine facilities, it cannot be treated as independent unit.

Appendix A

Screenshots of Kennecott's Greens Creek Mine simulation

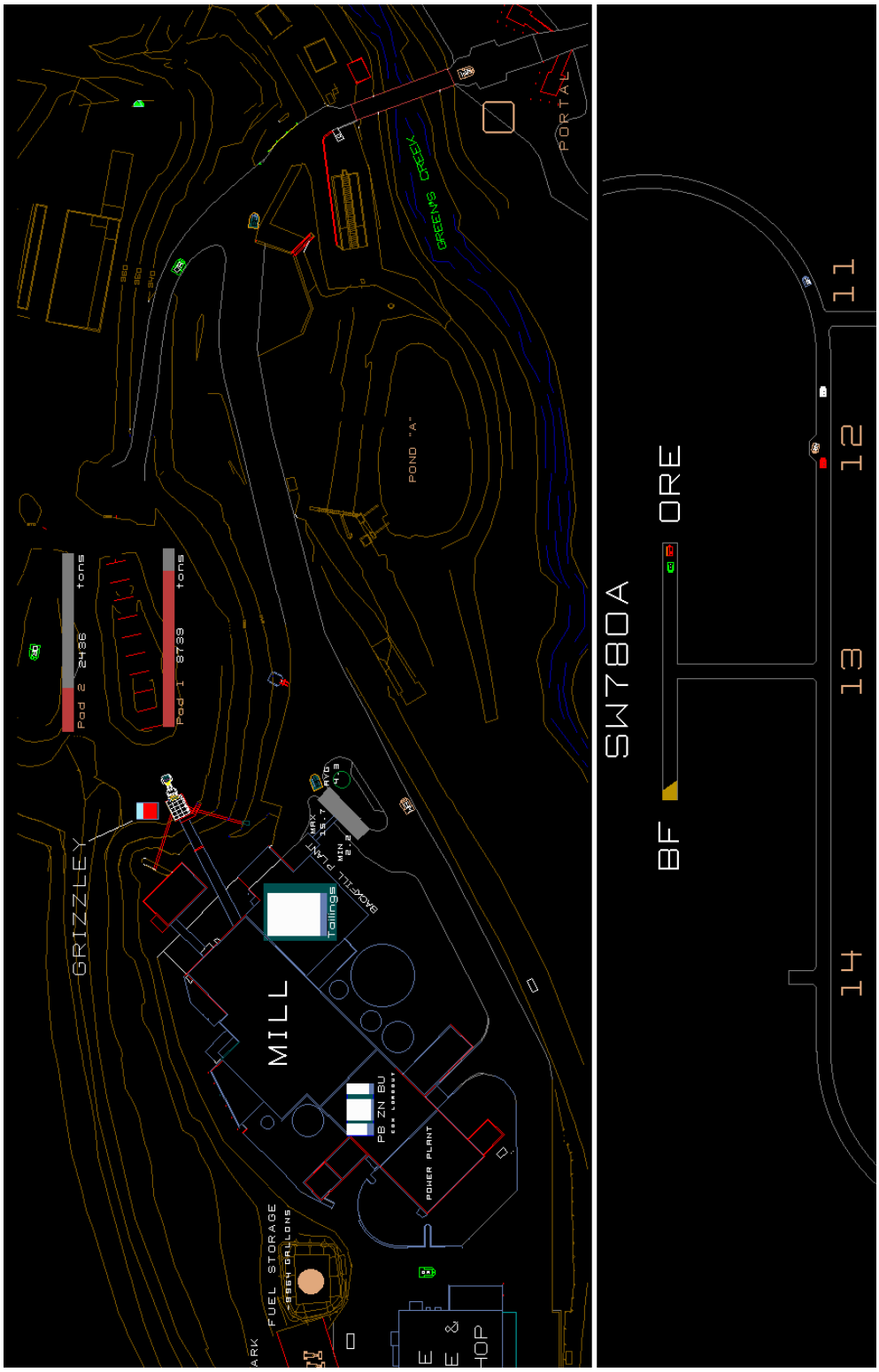
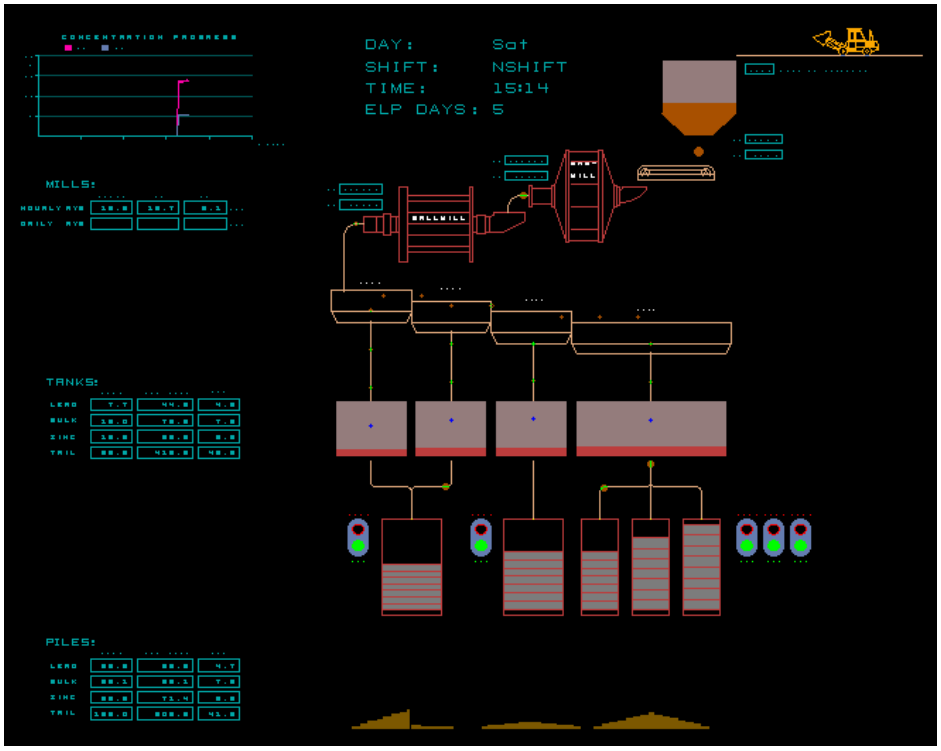


Figure A.1: Layout detail depicting the working cycle of an ore truck



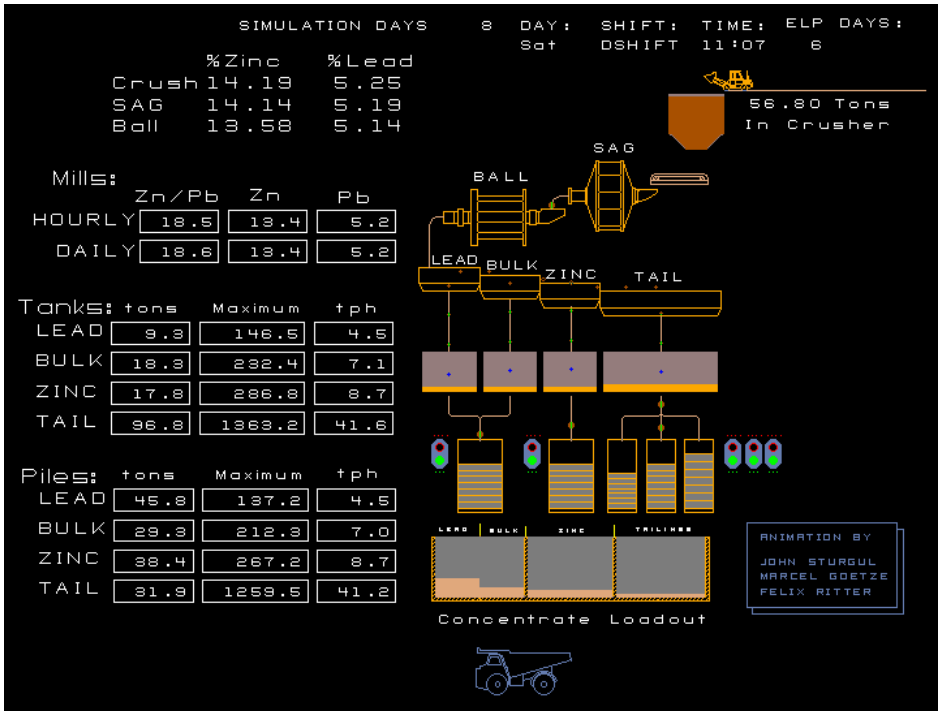


Figure A.2: The picture at the very top shows the first attempt to visualize the proceedings within the ore mill. This one also contains a plot of the concentration progress of lead and zinc after the ball–mill. However, in dialog with the customer the layout has been altered to correspond to aspects of an easier observation. The second figure shows the final design.

Bibliography

[Aka95] T. Akatsuka. Development of new techniques for combined continuous and discrete simulation, 1995.

[And90] R. Anderl. *CAD/CAM: Auf dem Weg zu einer branchenübergreifenden Integration*. Springer–Verlag, Berlin, 1990.

[Bau96] C. Bauer. *Nutzenorientierter Einsatz von Virtual Reality im Unternehmen*. Computerwoche Verlag GmbH, München, 1996.

[Ber92] Brian Berliner. CVS II: Parallelizing software development. Technical report, Prisma, Inc., Colorado Springs, 1992.

[FVU97] N. Fujii, J. Vaario, and K. Ueda. Potential field based simulation of self–organization in biological manufacturing systems. In *1st World Congress on Systems Simulation*, 1997.

[Göt98] M. Götze. Mining simulation. Internship report, "Otto von Guericke" University Magdeburg, 1998.

[HC] J.O. Henriksen and R.C. Crain. *GPSS/H Reference Manual*. Wolverine Corp., Annandale VA.

[Heu95] A. Heuer. *Datenbanken – Konzepte und Sprachen*. International Thomson Publishing Company, Bonn, 1995.

[HLA96] Interim definition of the DoD High Level Architecture for modeling and simulation. Interim definition, Defense Modeling & Simulation Office – American Department of Defense, `http://www.dmso.mil`, 1996.

[JDB95] Defense Modeling and Simulation Office. *Joint Data Base Elements for Modeling and Simulation Methodology Manual*, February 1995.

[Kuh93] A. Kuhn. ... in der Robotereinsatzplanung. In *Simulationsanwendungen in Produktion und Logistik*, volume 7 of *Fortschritte in der Simulationstechnik*. Vieweg Verlag, Braunschweig/Wiesbaden, 1993.

[Pro97] ProSTEP Produktdaten Technologie GmbH, Darmstadt. *PSstep_Caselib – Entwicklungsbibliothek für STEP kompatible Anwendungsprogramme*, 1997.

[Stu95] J. Sturgul. Simulation and animation - come of age in mining. *Engineer and Mining Journal*, October 1995.

[VFS+97] J. Vaario, N. Fujii, D. Scheffter, et al. Factory animation by self-organization principles. In *1st World Congress on Systems Simulation*, 1997.

[Wal94] J.C. Walther. *Verkehrssimulation in Transport- und Kommunikationsnetzen als Basis für Investitionsstrategien.* PhD thesis, University Fridericiana Karlsruhe, 1994. Department of economics.

[Wri90] E.A. Wright. *Open Pit Mine Design Models.* Trans Tech Publications, Clausthal-Zellerfeld, 1990.