Automation and Robotics in Construction XI
D.A. Chamberlain (Editor)
© 1994 Elsevier Science B.V. All rights reserved.

CONCPLANNER: AN AUTOMATED PROCESS PLANNING SYSTEM FOR CONCRETE CONSTRUCTION

By: Raghavan Kunigahalli[a] and Jeffrey S. Russell[b]

[a] Ph.D. Candidate, Dept. of Civ. & Env. Engrg., University of Wisconsin-Madison, Madison WI 53706.

[b] Asst. Prof., Dept. of Civ. & Env. Engrg., University of Wisconsin-Madison, Madison WI 53706.

Abstract: This paper presents **Concplanner**, a Computer Aided Process Planning system capable of generating *pictorial construction process plans* for automated concrete construction. **Concplanner** provides an efficient user-interface by adopting interactive computer graphic programming techniques. International Standards Organization computer graphic functional interface standard - PHIGS has been employed to: (1) obtain graphical input through logical input devices and (2) display *pictorial construction process plans*. An additional capability of generating an obstacle avoidance path plan for automated concrete placement has also been incorporated into the system. The output corresponding to obstacle avoidance path plan can be used to provide equipment-level instructions to Computer Numerically Controlled concrete placement systems.

1. INTRODUCTION

Operations, *processes*, and *work tasks* comprise three levels in the hierarchy of construction management that focus on field action. Work tasks are the *basic building blocks* of operations and processes and must be clear enough for a construction crew to grasp and visualize [1]. *Construction process planning* can be defined as the act of preparing detailed task-level instructions to execute a construction process. Large amounts of information that need to be processed at the construction task-level may, in general, result in *lengthy instructions* to: (1) construction workers and (2) controllers of automated equipment operating on job-sites. Computer Aided Process Planning (CAPP) systems for construction can: (1) shorten the instructions, (2) reduce the time required to generate a process plan, and (3) investigate a large number of alternatives using a combinatorial approach. In general, there are two basic methods that can be employed to develop CAPP systems: (1) a *variant process planning* method that retrieves an existing construction process plan for a process or operation identical to the one under investigation and (2) a *generative process planning* method that synthesizes the process information in order to create a process plan for a given situation automatically.

In this age of visual communication, people prefer reading less and viewing at a picture more. Object Oriented Programming (OOP) techniques allow operations of input devices such as "clicking a mouse button" and "selecting from a pull-down or pull-right menu" in order to generate different views of a given picture. Provisions for such user-friendly graphic interfaces reduces the amount of time required to train the construction engineers and technicians to effectively use emerging technologies. This paper describes a CAPP system entitled **Concplanner**. The system is linked to computer graphics, through OOP, in order to generate a *pictorial construction process plan* for concrete placement.

2. COMPUTER GRAPHIC PROGRAMMING PLATFORM

To-date, there are two computer graphic functional interface standards approved by International Standards Organization (ISO): (1) Graphic Kernel System (GKS) (ISO 7492) and (2) Programmer's Hierarchical Interactive Graphic System (PHIGS) (ISO/IEC 9592). While both Fortran and C-binding of GKS was adopted as standards in 1985, Fortran-binding of PHIGS was approved in 1988 and C-binding in 1991. These standards offer portability to many different computers using different operating system and window systems. PHIGS supports a hierarchical model, whereas, GKS uses an older model for graphics. Further, PHIGS is the first Application Programmer's Interface (API) supported by the M.I.T. X Consortium and is one of the most important APIs for 3-D graphics [2]. Both GKS and PHIGS standards were considered when selecting a graphic programming platform to generate pictorial construction process plans. PHIGS was selected because it is best suitable for hierarchical graphic models.

3. OVERVIEW OF PHIGS

PHIGS provides a link between application and the display device and enables manipulation of display device to deal with abstractions such as geometric objects and color. PEX is a X Window system protocol extension for 3-D graphics. Just as Xlib generates X protocol, PHIGS in the X environment generates PEX protocol. An efficient user-interface can be accomplished by using: (1) PHIGS to produce graphics and (2) Xlib or X toolkits such as Motif, XView, or OLIT to create other components such as menus and scroll bars.

Centralized Structure Store. Storage and manipulation of graphical and application-specific data are supported through a centralized hierarchical data structure, known as *Centralized Structure Store* (CSS). A fundamental entity of data is called a *structure element*. *Structure elements* are grouped together into units called *structures*. These *structures* are organized in directed acyclic graphs (DAG) in order to build *structure networks*.

Output Primitives. Primitives are basic building blocks for composing graphic images. PHIGS has 15 output primitives that can be grouped into the following: (1) line primitives, (2) area primitives, (3) text primitives, and (4) other primitives such as cell array and generalized drawing primitive. *Attributes* control a primitive's appearance such as color and style. These *attributes* can be either *individual* or *bundled* for a given workstation. The *attributes* can be set through an *aspect source flag*.

Input Devices. PHIGS provides six classes of *logical input devices* in order to enable application programs to effectively interact with the user. These *logical input devices* include: (1) locator, (2) stroke, (3) pick, (4) choice, (5) valuator, and (6) string. Each logical device's current value is called a *measure*. Striking the device's *trigger* causes the device to change its *measure*. The logical input devices can be used in any of the following three operating modes: (1) request, (2) sample, and (3) event. In the request mode, the program stops execution until the operator triggers the specified device. In the sample mode, application programs can obtain the *measure* of a device at any time during execution. The event mode of logical input devices results in an event queue similar to the X event queue.

Workstations. PHIGS is based on the concept of abstract graphical workstations that provide the logical interface for the application programs to control physical devices. The abstract workstation is capable of accepting input (or metafile input), output (or metafile output), or both. Each call to *open workstation* function available in PHIGS library associates a given *workstation identifier* with a generic *workstation type* and a *connection identifier*. The current state of each open workstation is kept in a workstation state list and can be obtained by calling the appropriate inquiry function. Occurrence of changes to the displayed picture can be controlled by setting a *display update state*. The display update state has two components: (1) deferral mode and (2) modification mode. The deferral mode determines when the workstation is updated and modification mode indicates whether the application program desires PHIGS to simulate the picture changes quickly.

Name Sets and Filters. *Highlighting*, *invisibility*, and *detectability* of output primitives can be manipulated by using *name sets* and *filters*. *Name set* assign names to individual primitives and *filters* are responsible for highlighting, making a primitive invisible, and setting a primitive as detectable by a pick device [2, 3].

4. COORDINATE SYSTEMS AND TRANSFORMATIONS IN PHIGS

The mappings of graphical output primitives and logical input values are performed by a series of transformations along following five different right-handed coordinate systems: (1) Modeling Coordinate (MC), (2) World Coordinate (WC), (3) View Reference Coordinate (VRC), (4) Normalized Projection Coordinate (NPC), and (5) Device Coordinate (DC). The series of transformations to the points applied during mappings is called *transformation pipeline*. Within the *transformation pipeline*, PHIGS uses homogeneous coordinates to represent points in Cartesian space and transformations are specified by a 4x4 or 3x3 homogeneous transformation matrix. There are three stages along the *transformation pipeline*: (1) modeling stage, (2) viewing stage, and (3) workstation stage.

The transformations from MC to WC pertain to the *modeling stage*. This stage transforms all primitives to a common coordinate system. Transformations from WC to NPC pertain to the *viewing stage*. This stage performs the following: (1) *view orientation* that orients the model with respect to the viewer and (2) *view mapping* to NPC. *View orientation* is specified by the following three parameters: (1) view reference point, (2) view plane normal, and (3) up vector. These three parameters define the VRC coordinate system. The *view mapping* selects: (1) view plane, (2) portion of the visible world coordinates, (3) type of projection (parallel or perspective), and (4) a 3-D projection viewport to NPC. The transformations from NPC to DC pertain to the *workstation stage*. This stage allows mapping of portions of the NPC cube to portions of workstation's display surface. The workstation transformation is specified by selecting: (1) a workstation window and (2) a workstation viewport.

5. DESCRIPTION OF CONCPLANNER

Concplanner generates a construction process plan for concrete placement using as input a wire-frame CAD model. In a wire-frame CAD model, there is no information on faces (such as rectangular slabs) and the adjacency relationships among them. Further, identification of portions that are *interior* and *exterior* to the given floor is not self-evident.

For instance, consider the example floor shown in Figure 1. Imagine the user constructing the picture on a computer terminal. Consider the following situation: the user clicks the mouse at point **x** and drags the mouse to point **y** and double clicks the mouse to complete the line corresponding to beam b-1. The points **x** and **y** are initially recorded in DC as vertices corresponding to the end points of the beam b-1. These end points in DC are transformed to WC and stored in a vertex table. As a next step, the user can proceed by constructing the line corresponding to any of the remaining 19 beams shown in Figure 1. The user can even pan and move over to the other corner to construct beam b-9. In such computer-aided drawing methods, there is no information to indicate that the beam b-1 is a boundary beam and it has concrete material only towards its right when moving from **x** to **y**. It is not even clear that beam b-1 is one of the edges that enclose the rectangular slab labeled **4**.

Concplanner obtains graphic input data corresponding to: (1) end points of beams in a given floor, (2) identification numbers of beams corresponding to the boundaries of the floor and obstacles (ordered clockwise), and (3) alphanumeric identifications of columns along with the central coordinate values. Following this, the **Concplanner** employs techniques from computational geometry to: (1) obtain intersections among edges, (2) obtain vertex to edge relationships, (3) form rectangular faces, (4) generate adjacency relationships among faces, and (5) associate columns to the beams supported by them. Face to face, vertex to face, and vertex to edge relationships can be readily printed by executing a print command. Formation of rectangular faces are simulated and *interior* and *exterior* (including obstacles) portions of the floor are displayed using different colors.

Upon completing the display of a picture that distinguishes the *interior* and *exterior* of the given floor, **Concplanner** prompts the user to select the *first rectangular slab where concrete is to be placed* by clicking the left mouse button anywhere in its interior. Upon clicking the mouse button, the selected rectangular slab flashes three times while the determination of a 'nearly optimal' sequence for concrete placement is found using combinatorial optimization techniques.

As an example, if the user first selects rectangular slab **1** shown in Figure 1, **Concplanner** generates the 'nearly optimal' sequence denoted as rectangular slabs **1** through **8**. A different sequence is generated when the user first selects rectangular slab **4** shown in Figure 1. The generated sequence is shown as **1'** through **8'** in Figure 2. **Concplanner** simulates the obtained sequence by hatching the rectangular portions one by one. As **Concplanner** supports off-line applications, a number of different possible sequences in automated concrete placement can be visualized prior to the actual placement process.

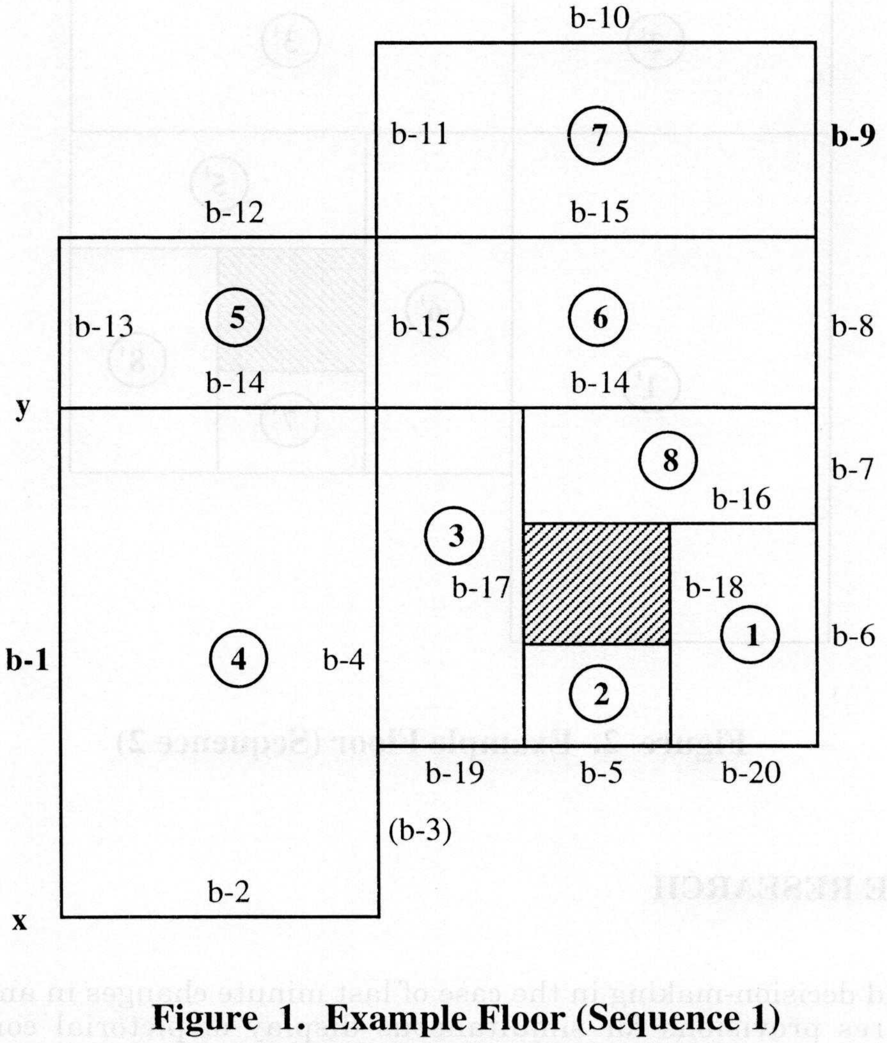

Figure 1. Example Floor (Sequence 1)

Additionally, **Concplanner** has the capability of generating a detailed obstacle avoidance path for the concrete placement pipe. The path of the placement pipe interior to rectangular slabs corresponds to a direction-parallel path. Paths between rectangular slabs along the 'nearly optimal' sequence correspond to a minimal rectilinear path. The user is prompted for additional input on placement pipe diameter in order to generate a detailed path plan. The detailed path plan generated by **Concplanner** can be used to provide equipment-level instructions to Computer Numerically Controlled (CNC) concrete placement systems.

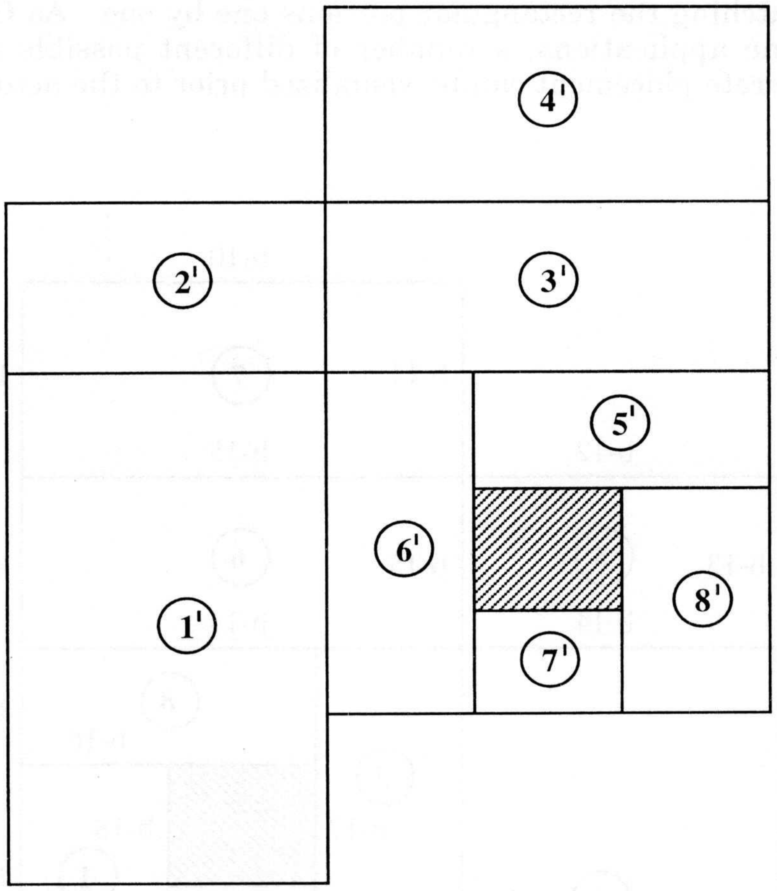

Figure 2. Example Floor (Sequence 2)

6. FUTURE RESEARCH

Rapid decision-making in the case of last minute changes in an execution plan requires provisions for simultaneous display of pictorial construction process plans at the job-site and possibly the corporate office. Extension of **Concplanner** to display process plans and interact with input devices across a network needs to be investigated. Further, safety precautions is vital to provide accident-free automated construction environment. A detailed ergonomic study related to concrete construction needs to be conducted in order to enhance **Concplanner's** ability to generate safety precautions.

Currently, the viewing angle is restricted to the direction perpendicular to the plan of a given floor. The system must be enhanced to allow several views of the same model simultaneously on separate projection viewport.

7. CONCLUSION

The need for construction process plans were discussed. A brief discussion on the two computer graphic standards approved by ISO was provided. A description of the selected computer graphic programming platform, PHIGS was provided. Five different coordinate systems defined in PHIGS and the various stages along its transformation pipeline were discussed. A description of **Concplanner**, a CAPP system for concrete construction, along with an example application was also provided.

8. ACKNOWLEDGEMENTS

The second author wishes to thank the National Science Foundation for Grant No. MSM-9058092, Presidential Young Investigator Award, for financial support of this effort.

9. REFERENCES

1. Halpin, D. W. and Riggs, L. S., *Planning and Analysis of Construction Operations*, John Wiley & Sons, (1992), New York, NY.

2. Gaskins, T., *PHIGS Programming Manual - 3D Programming in X: The Definitive Guides to the X Window System*, O'Reilly & Associates, Inc., (1992), Sebastopol, CA.

3. ANSI/ISO9592.1, *Computer Graphics - Programmer's Hierarchical Interactive Graphic System (PHIGS) Part I: Functional Description*, American National Standard Institute, (1989), New York, NY.