THE APPLICATION OF INTERACTIVE GRAPHICS TO LARGE TIME-DEPENDENT HYDRODYNAMICS PROBLEMS*

Fred Gama-Lobo and Lynn D. Maas
Los Alamos Scientific Laboratory

Introduction

This paper is a companion to a movie produced at LASL entitled "Interactive Graphics at Los Alamos Scientific Laboratory". The intent of the paper is to complement the movie. The movie presents the actual graphics terminal and the functions performed on it much better than any written description could possibly convey. However, the movie compresses and simplifies the processes being presented because the pace of a movie prevents detailed description. This paper attempts to put in perspective the complexity of the application code and the complexity of the interaction that is possible, which may not be apparent from the simple problem run as an example in the movie and the few simple examples of interaction presented in the movie. The paper, by itself, does not have as much of an impact, however, without the visual examples presented in the movie.

Interactive Graphics System Description

The interactive refresh graphics system in use at LASL consists of a Sanders Associates, Inc. Model 960 display indicator, a Varian 620i minicomputer, and two Control Data Corporation (CDC) 7600 computers. The minicomputer controls two CRT displays, handles input devices, and communicates with the CDC 7600 computers on which the user programs run (see Figure 1).

A terminal consists of an alphanumeric keyboard, 16 system control keys, 16 lighted user function keys, a data tablet, a light pen, and a CRT. The minicomputer currently is supporting two terminals. One of these terminals has a single-color CRT with a display area of 14 inches square (1024 by 1024 resolution) while the other terminal has a four-color CRT (CPS, Inc. Penetration CRT) with a display area of 10 inches square (1024 by 1024 resolution).

*This work was performed under USERDA Contract W-7405-Eng. 36.

The minicomputer communicates with the two CDC 7600 host computers via a multiplexed IO path at the rate of 3.2 megabits per second. The CDC 7600 host computers are running a mono-program batch operating system with modifications to allow interaction with the graphics system. Interaction is achieved by means of a high job priority and a disk rollin-rollout capability in the host computer.

This system was designed to provide interactive graphics for large, typically long-running programs that had previously been restricted to batch operation.

Application Code

One code that was adapted to run on this system is a two-dimensional Lagrangian hydrodynamics code. The problems run by this code are set up as a two-dimensional matrix of points. These points define a mesh of quadrilateral zones that describe the geometry of the problem. The code carries the geometric information and all the physical attributes of the problem in doubly-dimensioned arrays. It carries two coordinates and two components of velocity for each mesh point and it carries a material identification, mass, density, temperature, pressure and other physical quantities for each quadrilateral zone. The indices to these arrays are called K and L by convention. For a constant K, points sequentially indexed by L are connected by line segments and a sequence of such connected points is called a K line. For a constant L, points sequentially indexed by K are connected by line segments and a sequence of such connected points is called an L line. The intersections of K and L lines form the quadrilateral zones. By convention, the zone quantity arrays indexed by K and L refer to the quadrilateral connecting the points (K,L) , (K-1,L) , (K-1,L-1) , and (K,L-1).

The code computes the motion of the problem in time by breaking the problem up into discrete time steps. The following is a simplified description of the computational cycle for each time step. First, the code establishes a time step for the cycle based on various stability criteria. The code then uses numerical difference equations based on analytic equations describing the physical phenomena occuring in the problem to determine the motion of each mesh point for the cycle. This is essentially done by differencing the pressures of the zones surrounding a point to determine the change in velocity for the point during that cycle and then multiplying the velocity by the time step to determine the change in position of the point. The mass of each zone does not change with time, so the new volume of each zone is used to compute a new density. Tabular equations of state are then used to get a new pressure for each zone. The code then goes through the same cycle for the next time step.

Rezoning

An annoying problem with Lagrangian hydrodynamics codes is mesh tangling. As was mentioned before, the motion of each point is calculated independently. One would hope that if the equations used are correct, all the zones would stay regular. However, the complex motions of the mesh can cause some of the zones to get very distorted. These distortions can sometimes cause problems, such as zones getting artificially high temperatures and pressures because the zones are unrealistically compressed or the distorted zones causing the problem to run at a lower than normal time step, thereby requiring more computer time to complete the problem. There are also unphysical distortions that occur when a mesh point somehow gets a spurious velocity that causes it to cross over into a zone outside its surrounding zones. The reason such problems occur is not always known, but it is probably because of trying to describe a continuous medium with a finite number of zones.

The remedy to these problems is rezoning. This is done by moving points to reduce the distortions. The new mesh configuration is then mapped onto the old mesh to redistribute the physical quantities, such as mass and energy, based on the intersections of new zones with old zones.

Interactive Techniques Used in the Code

Rezoning

The most significant application of interactive graphics in the code is in rezoning. The rezoning process deals almost entirely with graphical data and is done very naturally at the graphics terminal. The easiest way to discuss the value of interactive graphics in this application is to compare the rezoning process prior to the availability of interactive graphics with the current techniques.

The old rezoning process was as follows. First, a drawing of the mesh was obtained by running a restart dump tape through a post-processing code which generated a plot on a Calcomp plotter. This mesh plot was then examined to determine the area where tangling occured. At this point, it was often necessary to get another mesh plot of an enlarged window around the area of tangling, because such tangling usually occurs where zones are being crushed and points are so close together that they are not resolved in a drawing of the full mesh. In order to define such a window, it was necessary to look up coordinates of points in a listing, sketch the desired window on the drawing, and measure relative distances to the known points to determine the coordinates of the window limits.

At the graphics terminal, a series of windows can be examined in succession and the area of mesh tangling can be quickly isolated. No coordinates need be known because the user can define the window he wants from the full mesh display on the screen by using the data tablet to point to the positions of the window limits. If the resulting window is still too big to resolve the tangled lines, he can continue defining smaller windows until he does resolve the tangled lines.

Rezoning is accomplished by moving mesh points to reduce distortions. There are several input commands to the rezone code to do this. The most natural method is to move a point by giving it a new set of coordinates. With card input, the coordinates have to be punched on a card. Determining the coordinates needed to move a point can be complicated if one has to extract this information from a mesh drawing. It is easy enough to mark the new position of the point on the drawing, but it is then necessary to measure the components of the distance of that point to an existing point in the problem, look up the coordinates of that point in a listing, scale the measured components from the drawing to the problem coordinate system, and then add them to the coordinates of the existing point. Since this is so cumbersome, people tended to use other available commands that do such things as equally space lines, straighten lines, and map the proportional spacing along one line onto another. The command to equally space a line, for instance, needs only the indices of the two end points of the line specified.

At the graphics terminal, it is very easy to move a point. The user points to a mesh point on the display and then points to the new location, using the data tablet. The display then changes and the mesh point moves instantly to its new location. Commands such as equal space are also simplified. Determining indices of points from a mesh drawing can be a nuisance, too, when dealing with a large problem having many points. At the graphics terminal, the user points to the end points of the line to be equally spaced, using the tablet. Again, the display changes instantly to show the results of the equal space command.

This brings up another advantage of interactive graphics. The entire rezoning process takes place sequentially with each step producing a new picture. In the batch mode, the entire rezone would have to be predetermined prior to submitting a rezone run. This was usually done by sketching on the mesh drawing. If the rezone was complicated, then the drawing would become very confusing, thereby necessitating several batch submitals of rezone runs, each requiring a post-processing run to generate mesh drawings. It was not unusual for a rezone to consume several days, and therefore people avoided doing them.

There is a class of problems that will not run without a large number of rezones. Some problems run recently have required some rezoning about every 5 minutes of CP time, with the problem running a total of 5 hours of CP time. Such problems were usually not run prior to the interactive graphics capability, and if they were, they would have taken several months. Now they can be done in four or five terminal sessions of approximately two hours each.

Monitoring

The other key interactive feature of the code is the capability of monitoring the execution of the code and interrupting at any time to rezone. The execution is initiated by typing RUN at the keyboard. While the problem is being monitored, successive pictures are displayed on the screen at time intervals specified by the user. In this mode, the code remains resident in the host computer and just continues running. The user can stop the run at any time by hitting a fuction key. The user can then examine the mesh and various problem values printed on the screen and he has the option of rezoning or continuing to run the problem. This capability is essential to give the user total control of the calculation at the terminal.

In the initial implementation of interactive graphics in this code, the rezone package was a separate code. It ran interactively, while the main computational code still ran in the batch mode. There was an immediate payoff in the simplification of rezoning, but for problems requiring 50 rezones, that meant submitting 50 batch computational runs and scheduling 50 terminal sessions to do the rezoning.

This was not the only drawback. The way the computational code communicated with the rezone code was by restart dumps on tape taken at time intervals specified by the user. Quite often these dumps were not available at the best times for rezoning. It was not unusual to get a batch run back in which the mesh looked good for one time dump and was so badly distorted on the next dump that the rezoning required became very complicated. The monitoring capability lets the user pick the optimum rezoning times. He can do this visually by looking at the mesh or by examining various quantities printed on the screen which may indicate that there are problems arising in the calculation.

It is this monitoring function which necessitates the link to a machine with the great computational speed and capacity of a CDC 7600. For typical problems, the meshes range from 5000 to 10000 zones and describe very complicated geometries. A large memory is required to hold the many physical quantities stored for the mesh. Also, the numerical difference equations used by the code to describe the physical processes being studied with the aid of this code are very complex and require the computational speed of the fastest computers currently available to run problems in a reasonable amount of time. Typical problems can consume from 5 to 10 hours of CP time. The rezoning, by itself, could be done by a much slower computer, but the capability of total interactive control of a calculation that the monitoring function, along with the rezoning, provides, absolutely requires that the graphics terminal be linked to the fast computer.

The batch operating system on the host computer turns out to be an advantage for this application. This class of code generally does not compete well in a time-sharing environment. During a terminal session, a user will typically consume an hour of CP time, consisting of five minute chunks of computational time separated by rezones. For the user to make efficient use of his time at the terminal, those 5 minute computational chunks requiring the full capacity of the host should be executed without interruption. On this system, that is possible. The number of possible terminals is severely limited on such a system, of course, but in a computing environment consisting of many large, long-running codes, the number of terminals must be limited.

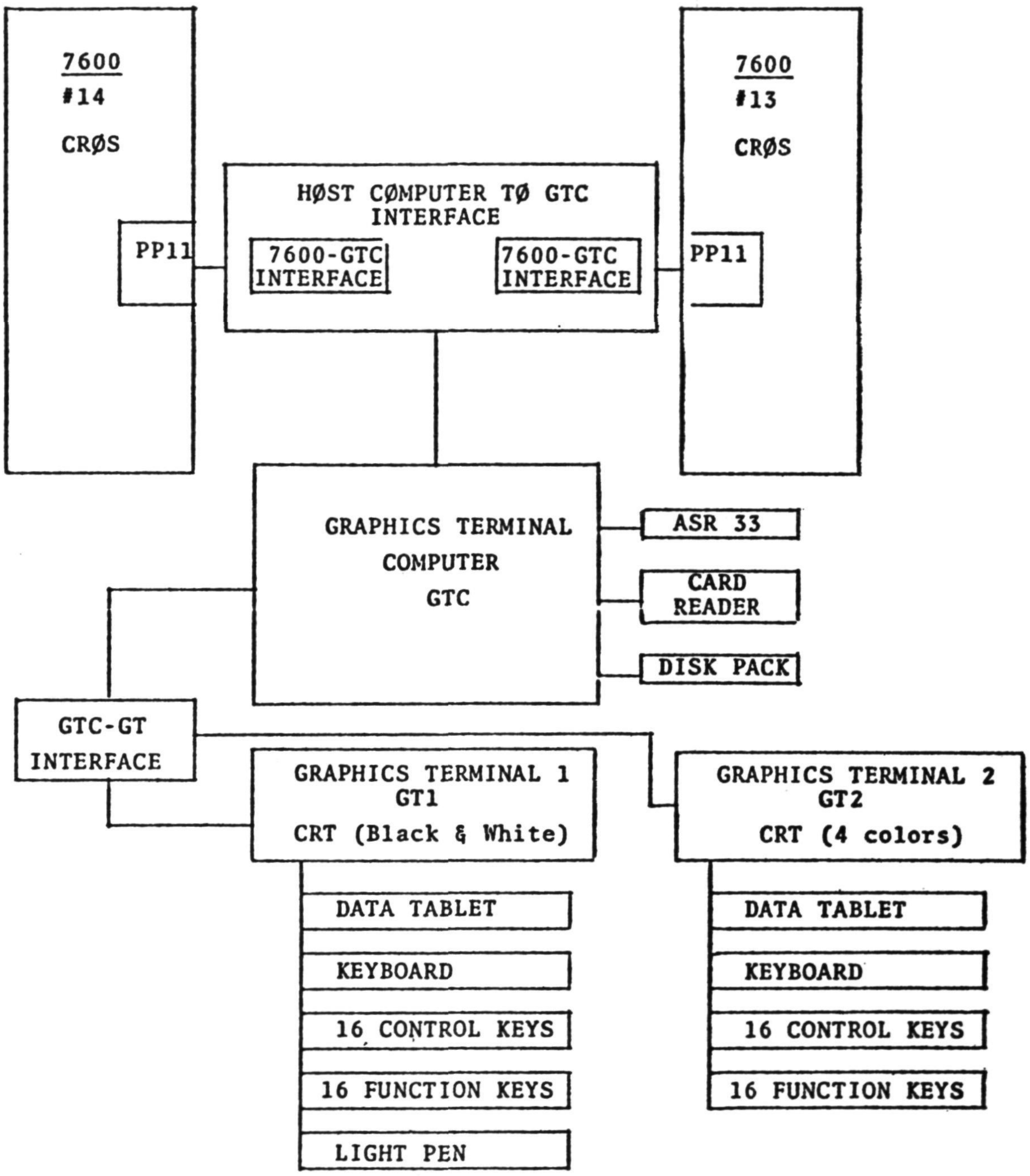

Figure 1
Hardware Configuration of LASL Interactive Graphics System