An Overview of the Graphic Elements System

The Graphic Elements system is based on the interaction between an application program (or other enclosing software entity, such as a pane or view in a class library system), a display controller which deals with groups of graphical entities (GEWorlds or worlds), and a number of graphical entities (Graphic Elements or elements). The relationships between the parts of the Graphic Elements system are shown in Diagram 1.

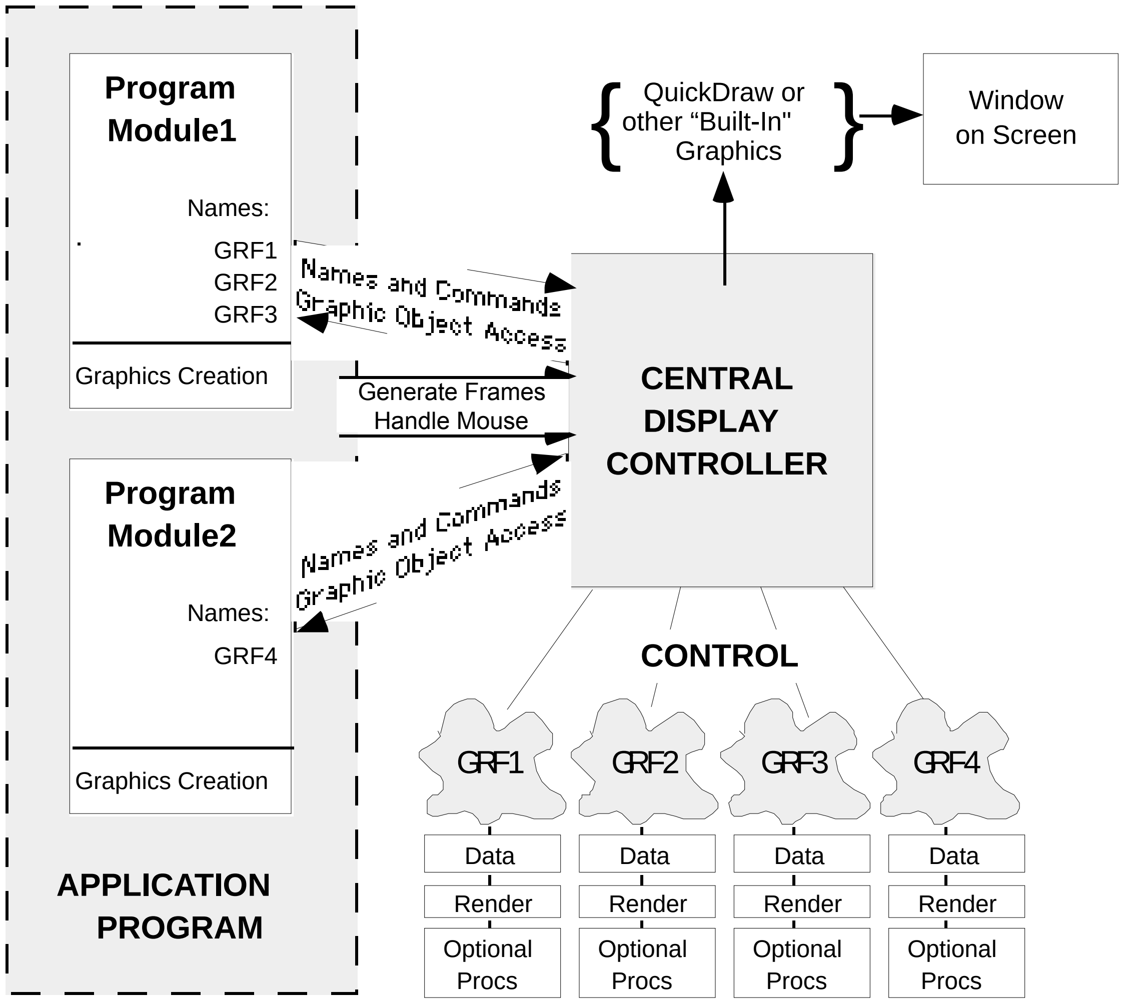

Diagram 1. The relationships between Application Program, Display Controller, and Graphic Elements.

The application program calls the display controller to create new Graphic Elements and GEWorlds to store them in, to generate updated images and display them on the screen, and to give the elements in a world the opportunity to respond to user actions. Through the display controller, the application can also access individual Graphic Elements and manipulate them explicitly

The display controller manages the memory allocated for Graphic Elements and offscreen work areas, and provides access to the data structures of individual elements. It calls individual Graphic Elements to perform their periodic functions or to interact with each other or the user, calls individual elements to render themselves as needed, and “projects” updated frames onscreen as demanded by the application program.

Each related group of Graphic Elements handled by the display controller is one GEWorld. A Graphic Elements world is a rectangle, within a window, for which the Graphic Elements system handles all graphic operations.

One world may contain any number of Graphic Elements (limited only by memory and processor speed), which may lie on up to 32,767 “planes” within this world. Elements on higher-numbered planes are drawn “in front of” elements on lower-numbered planes.

Each Graphic Element in a GEWorld is, at a minimum, responsible for rendering itself, on demand from the display controller, into a graphical environment provided by the display controller. Optionally, it may do any or all of the following: 1) Change its appearance periodically, for example by moving or changing frames; 2) interact with other Graphic Elements (collision); and 3) interact with the user (by tracking mouse movements and executing an action procedure in response to the user releasing the mouse button).

Anything that can be drawn on the screen can be a Graphic Element. Each Graphic Element is identified by a unique 4-character “name.”

Display Controller Interactions

The display controller interacts with the display "hardware" (actually, with the combination of hardware and system-level graphics facilities), with the application program, and with the individual Graphic Elements. These interactions are shown in Diagram 2.

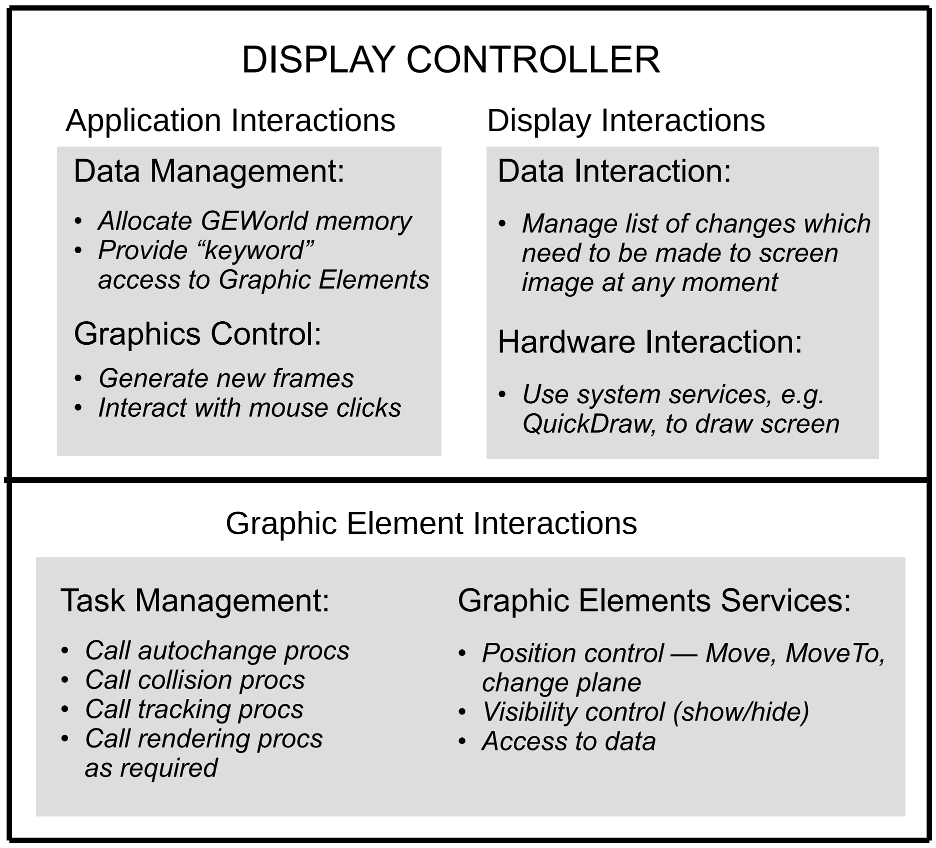
DISPLAY CONTROLLER
Application Interactions
Data Management:
• Allocate GEWorld memory
• Provide "keyword" access to Graphic Elements
Graphics Control:
• Generate new frames
• Interact with mouse clicks
Display Interactions
Data Interaction:
• Manage list of changes which need to be made to screen image at any moment
Hardware Interaction:
• Use system services, e.g. QuickDraw, to draw screen
Graphic Element Interactions
Task Management:
• Call autochange procs
• Call collision procs
• Call tracking procs
• Call rendering procs as required
Graphic Elements Services:
• Position control — Move, MoveTo, change plane
• Visibility control (show/hide)
• Access to data

Diagram 2. Interactions of Display Controller with display hardware, application program, and individual Graphic Elements.

The application program deals directly with the display controller to create and maintain Graphic Elements worlds. The bulk of this interaction takes place at a few well-defined times. The application creates GEWorlds and installs them into a window. It may activate and deactivate them, and set their minimum update intervals. Through the display controller, it gives them regular opportunities to update their appearance and to handle user actions within their screen areas. The application may also call the display controller to gain access to the data record of any individual Graphic Element, and may manipulate elements explicitly at any time. Finally, the application program uses the display controller to dispose of GEworlds and the elements they contain when it has finished using them.

The display controller handles all interactions with the graphics hardware. It maintains a list of all changes which have been made since it generated the last frame, and calls the Graphic Elements affected by these changes to draw the required portions of themselves. The display controller then uses services provided by the operating system to transfer the results of these changes to the computer screen.

The display controller interacts with some or all of the individual Graphic Elements in a GEWorld during the generation of each graphics frame. Each element may have an “autochange” procedure, which is called periodically by the display controller to give that element an opportunity to move, change frames, or perform any other time-based task. Each element may also have a collision procedure, called by the display controller whenever that element comes in contact with an element in its “collision plane.” Each element must have a rendering procedure, which draws some or all of the element as required by the display controller. Finally, the display controller acts as an intermediary in handling user interactions: when the application passes a mouseDown event to the display controller, it finds the Graphic Element which handles events in that area of the screen, and passes the event along to that element's mouse-tracking procedure.

The display controller provides the same services to individual Graphic Elements that it provides to the application program. Individual elements call the display controller to change their position, their plane, or their visibility. They may also call the display controller to gain access to their own data records or those of other Graphic Elements.

Graphic Element Interactions

Every interaction in the Graphic Elements system results in an action by, or on, an individual Graphic Element. Graphic Elements are “smart” graphical objects which may change their appearance, position, or other attributes in response to any, all, or none of three kinds of external events. These events include: 1) the passage of time, 2) collision with another Graphic Element, and 3) being “clicked” on by the user. All of these events are detected by the display controller and passed on to the appropriate Graphic Element.

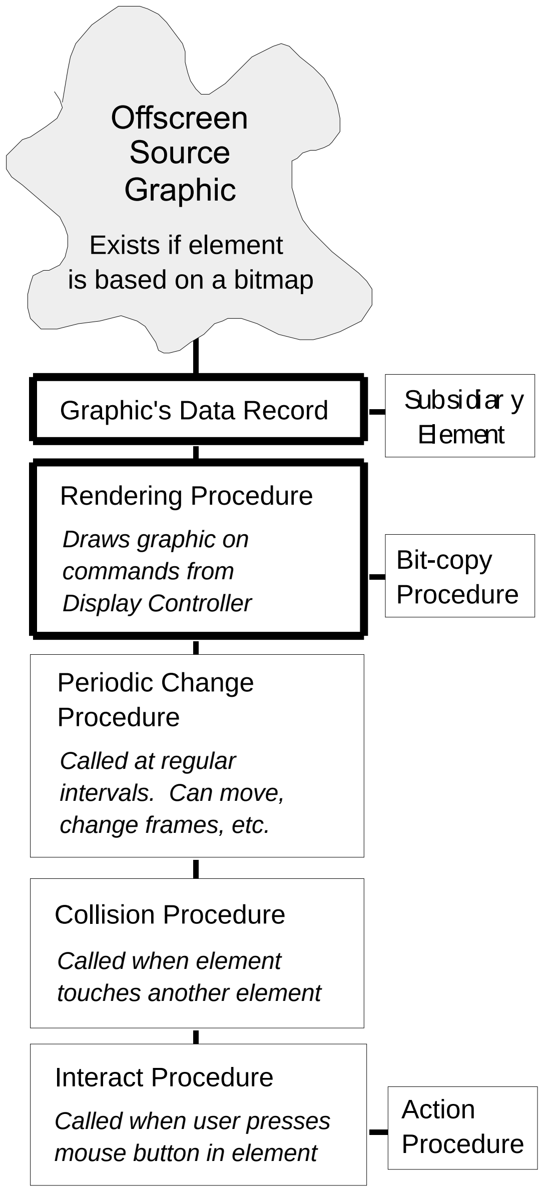
Offscreen Source Graphic
Exists if element is based on a bitmap
Graphic's Data Record
Subsidiary Element
Rendering Procedure
Draws graphic on commands from Display Controller
Bit-copy Procedure
Periodic Change Procedure
Called at regular intervals. Can move, change frames, etc.
Collision Procedure
Called when element touches another element
Interact Procedure
Called when user presses mouse button in element
Action Procedure

Diagram 3. An individual Graphic Element. Required components are outlined in bold.

As Diagram 3 above shows, a Graphic Element has only two essential components: a data record, and a rendering procedure. Its data record includes all the information needed by the display controller to access and manipulate it. This data record contains the element's identification (a four-character “name” used to gain access to the element), its location (a rectangle in the coordinates of its GEWorld), and its plane (a number between 0 and 32,767 representing its “height” above the background), along with other information. An element's data record may also include pointers to “master” and “slave” Graphic Elements. A “slave” element automatically maintains its horizontal and vertical distance from its “master” element. Since this “slave” can also have a “slave” element, the application program can easily construct “chains” of graphics that move together.

A Graphic Element's rendering procedure is a procedure which is capable of drawing all or part of the element at a location specified by the display controller, into a graphical environment provided by the display controller. This rendering procedure may use any appropriate method to do its drawing. In many cases, the rendering operation will involve copying a pixel array from one area of memory to another. A Graphic Element may optionally include a specialized routine to perform this copying.

The real flexibility and utility of Graphic Elements derive from their optional components.

Each Graphic Element may have an autochange procedure, in conjunction with a change interval. This procedure will be called by the display controller, at appropriate intervals, during frame-generation cycles for the world containing the element. It can change the element's appearance and position, or the appearance and position of any other element, as desired.

Each Graphic Element may have a collision procedure, together with a collision plane. The display controller will call this procedure any time the element comes in contact with another Graphic Element on its collision plane.

Finally, each Graphic Element may have an interaction procedure, called by the display controller whenever the user presses the mouse button while the cursor is within the screen area of that element. This interaction procedure will normally track the user's actions with the mouse, and may call an optional action procedure, for example as the user moves the mouse or when the mouse button is released within its area.

Graphic Elements—Method of Use

The Graphic Elements system is designed to be an open system. It is as free as possible of all artificial restrictions on data access and sequence of events. Thus the following constitutes only a recommendation, a structured framework for using Graphic Elements which has been found to be convenient, easy to understand and maintain, and portable. Note that the term “application program,” as used in this document, also includes such software entities as the panes and views of various class libraries. In general, it denotes any software module which creates, uses, and disposes of Graphic Elements worlds.

First, divide the Graphic Elements to be used into “scenes,” groups of related or interacting elements. For example, all components of the background could be placed into one scene.

Each scene should include all of the “optional” procedures for elements within that scene, for example autochange procedures and collision procedures. Each scene should also include a function to intialize all the scene's elements by creating them, installing their autochange and collision procedures, installing their subsidiary elements as required, etc. This function is called from the application program, and should return either a Boolean or an error code so that the application can tell whether the scene has been successfully initialized.

During its own initialization phase, the application program creates a window, installs a GEWorld of the appropriate size in it, and calls each of the scene initialization functions. If all the initialization functions execute without errors, the application activates the GEWorld.

From this point on, the Graphic Elements system is almost automatic. Once each time through its main event loop, the application program calls the display controller to generate a new frame if required. When the application receives a mouseDown event in the area covered by the GEWorld, it passes the event to the display controller for eventual action by one of the Graphic Elements in that world.

This same basic procedure may be replicated as necessary for software systems which require multiple windows and/or multiple Graphic Elements worlds.