ヒューマンインタフェース 41－4
（1992. 3. 2）

SYNTHETIC MEDIA ARCHITECTURE AS AN OPEN PLATFORM FOR THE NEXT-GENERATION INFORMATION PROCESSING

Yuzuru Tanaka

Electrical Engineering Department, Hokkaido University

N-13, W-8, Sapporo, 060 JAPAN

Computers as meta-media should provide us with an overall integrated environment for our intellectual activities. The scope of our activities changes with respect to time. Therefore we must define such an overall environment as an open system that allows addition of new tools at any time. To achieve this goal, we need a unified framework for the modeling, the presentation, the synthesis, the sharing, the circulation / publication, and the management of multimedia documents, system-provided functions, and application programs. In 1989, we proposed the IntelligentPad system as a hypermedia system with a unified framework for the modeling, the presentation, the synthesis, and the management of multimedia documents, system-provided functions, and application programs. It represents everything as a pad or a sheet of paper. Pads are all persistent. Different pads are associated with different functions. The pasting of pads on another pad defines a new pad that has both an arbitrary layout of fields and a new function composed of the constituent pads. Here in this paper, we will show some extensions for the provision of a shared pad space, the coordination of interactions, and the integrated management of pads, and propose the IntelligentPad architecture as an overall integrated open platform for the next-generation information processing systems.

次世代情報処理のプラットフォームとしての シンセティック・メディア・アーキテクチャ

田中 譲

北海道大学工学部電気工学科

メタメディアとしてのコンピュータは我々の知的活動を包括的に支援する環境を提供する。我々の活動の対象範囲は時とともに変化するので、それに応じて新しい道具が自在に加えられるオープンなシステムとして支援環境を定義しなければならない。そのためには、マルチメディア文書とシステム提供機能、それに各種応用プログラムのすべてに対して、モデリング、プレゼンテーション、合成、共有、流通、管理のすべてを通じて統一的な取り扱いが可能な表現と操作の枠組みが必要である。IntelligentPadは、モデリング、プレゼンテーション、合成、管理に関する統一的枠組みをもったハイパーメディア・システムとして1989年にていあんされた。このシステムはすべてのものをパッド、つまり一枚の紙として表現する。パッドはすべて永続的である。異なる種類のパッドは異なる機能を持つ。パッドにパッドを貼ることにより、レイアウトのデザインと機能の合成を行うことができる。この報告書では、共有空間の提供、スクリプト・プログラミング、統合管理の機能の追加拡充について述べ、IntelligentPadを次世代情報処理の包括的開放型統合プラットフォームとして提案する。

1. MIND TOOLS AND MEDIA

1.1. Mind Tools

Computers today are extending their definition from what computes expressions to what augments our intellect. The augmentation of human intellect might have two possible directions, i.e., 1) its automation and 2) the provision of useful tools. Our primary concerns in the automation are (1) how to declaratively describe the problem without losing consistency with what to solve, and (2) how to logically translate the given specification to its executable form. Our primary concerns in mind tool systems however are (1) how to intuitively capture the functions and the usage of each tool, and (2) how to construct a new tool from already existing ones. Some psychologists believe that a shape of a cup affords us how to handle it. This effect is referred to as 'affordance' [1]. The formats of documents and the shapes of deskwork tools also afford us how to manipulate and manage them.

The use of affordance implies the use of appropriate metaphors in man-machine interfaces. Before 1980, computers adopted a typewriter metaphor for their man-machine interfaces. Then in the 1980s, the metaphor of published documents prevailed among DTP systems. Then came the metaphor of tools to implement various OA tools on computers. Now computers come to incorporate multimedia facilities through the metaphors of audiovisual devices. Some researchers are developing systems with interface agents where a metaphorical secretary or agent lives in a computer and serves his master. Such a system may have more than one agents to form their society. It may also have urban facilities such as schools, libraries, department stores, and transportation facilities to form a metaphorical city. In this city, residents are either mechanical agents or other users connected to this system through computer networks.

Here in this paper, we are only concerned with the metaphors of documents, tools and audiovisual devices. Metaphors of these kinds are referred to by media metaphors.

1.2 Media Metaphors

Informations, when we receive, process, and send them, cannot exist without their media. Media have added various functions to information. Among them are the functions to visualize, to record, to archive, to transport, to duplicate, and to broadcast information. Media on computers have added more functions to information such as editing, programming information, associating information with each other, and interacting with information.

Media objects are classified into two categories: (1) static media objects, and (2) dynamic media objects. Text documents are static media objects, while documents with supporting tools behind them are dynamic media objects. The IntelligentPad System developed in Hokkaido University represents all the media objects as dynamic media objects, and assumes each of them to have a rectangular planer appearance [2]. It calls such an object a pad.

The adoption of media metaphors started in early 80's with the DTP. Around the mid 80's, AV storage media such as CD-ROM, LD, and LD-ROM were interfaced to computers, and video boards enabled display windows to show video clips with sounds. Complex documents were further extended to have embedded video clips. Then people began to seek a way to treat different types of components in a uniform way. This encouraged the object oriented system design, which allowed us to embed programs in complex documents. Such documents became dynamic media objects.

The object oriented programming allows two different styles of programming. The refinement-based programming defines classes and their property inheritance hierarchy, while the synthetic programming defines a set of object instances as primitive objects. In the latter, programmers can combine several primitives to define a new composite object. This programming style is also called the synthetic programming. Around 1986, the object-oriented modeling of media systems encouraged our research group to apply the synthetic programming style to the design of dynamic media systems. This led to our proposal of a synthetic media architecture, and resulted in the development of The IntelligentPad System.

2. SYNTHETIC MEDIA

2.1 Media Environment

Computers as meta-media provide varieties of information media and tools, which constitute our environment for intellectual activities. Such an environment should be an overall integrated one. Different look-and-feels for different media would require different treatment of different media and hence frequent conversions among them, which seriously disturbs our thinking. Therefore they should have the same look-and-feel. Since the scope of our activities changes with respect to time, we must define such an overall environment as an open system that allows us to add new types of information and tools at any time in the future. Such an environment requires the following four types of integration:

(1) **Homogeneity** Integration of documents and tools through their homogeneous representation.

(2) **Integrity** Logical integration of media objects by keeping the consistency of mutually related information.

(3) **Compoundability** Visual integration of multimedia objects by arranging them to define a single complex document.

(4) **Composability** Integration of various functions through the flexible combination of media objects to compose a new media object with a new function.

Todays hypermedia systems and integrated office tool systems are however closed systems. They provide no means for their users either to assimilate new tools developed elsewhere into their integrated environments or to compose new tools from existing ones. HyperCard, for example, has encouraged the development of lots of stackwares. Its users, however, cannot combine two different stackwares to compose a new one. Though NeXT's Interface Builder allows us to compose new tools, it distinguishes toolkit objects from windows. It represents some media objects as windows and the others as toolkit objects. It lacks the homogeneity of the representation.

One promising solution to this problem might be an object-oriented open architecture proposed by Tsichritzis [3]. He proposed this for software development systems. When applied to multimedia platform systems, however, it means a new architectural paradigm based on the following four principles:

(1) **generic toolkit**

Generic definition of primitive dynamic media objects.

(2) **synthetic programming**

Synthetic programming for the composition of new dynamic media objects from primitive ones.

(3) **open platform**

Standardization of platforms to make every primitive media object executable on as many machines as possible.

(4) **integrated management**

The integrated management of all kinds of media objects.

We call such systems satisfying these principles synthetic media systems. The first three principles would bring a new portable representation form for both documents and application programs. Naffah called this form the reactive information [4]. As he pointed out, it allows us to maintain information longer in soft form, than in hardcopy or paper. This facilitates the distribution of various multimedia documents and application programs, stimulates their production and usage as dynamic media objects, and opens a new media business market. Wide-area computer networks will offer good infrastructures for such markets.

2.2 Media Base

The increasing number of primitive and composite media objects will make it difficult for us to select appropriate objects to compose what we want. Its solution requires a database for the integrated management of all kinds of media objects. We call such databases media bases. Media bases have to manage a large amount of different object types, while the conventional databases as well as ADT and object-oriented databases can only deal with a large amount of objects that can be classified into a small number of different types. Implementation of a media base requires the development of new management and retrieval methods.

In media bases, the modeling of each object consists of three parts, i.e., its model, view and controller. Its model defines its state and behavior. Its view defines its appearance, while its controller specifies its reaction against user manipulation. Conventional databases store for each object only its state, while abstract-data-type databases and object-oriented databases store for each object both its state and behavior. Media bases store each object not only with its state and behavior but also with its appearance and reactions.

3. The IntelligentPad ARCHITECTURE

3.1 Design Philosophy

The IntelligentPad system is a media-synthesizing toolkit system based on the object orientation paradigm. Figure 3.1 shows a display snapshot of the system. It represents everything as a pad or a sheet of paper. Pads are all persistent. Different pads are associated with different functions such as word processing, image editing, line drawing, tabulation, graph drawing etc. Pasting of pads on another pad defines a new pad that has both an arbitrary layout of fields and a new function composed of the constituent pads. IntelligentPad initially provides a set of primitive pads. Its users can define new pads by pasting existing pads together.

IntelligentPad provides the following pad operations: (1) **open/close** of a pad, (2) **property change** of a pad, (3) **registration** and **naming** of a pad, (4) **copying/deleting** of a pad, (5) **moving** of a pad, (6) **save/load** of a pad, (7) **paste/peel** of a pad, (8) **tree representation** of a composition structure, and (9) **printing** of a pad. Each closed pad has an iconic representation. Each icon may have an arbitrary size and an arbitrary picture. It may be transparent. A click on an icon opens it. Each pad has a property sheet on which we can specify its

properties. Among them are its size, the shape of its border, and its shading and pattern. We can paste a pad on another pad, and peel a subpad off a pad. These operations are also applicable to closed pads. A peeled subpad keeps its state before the separation, but breaks the functional composition. Once a closed pad P_2 is pasted on P_1, the dependency between them is kept alive even after we open P_2. We may open P_2 at any location on the screen.

3.2 Basic Architecture

Each pad is implemented as an MVC (Model, View, and Controller) triple. Each pad has an update-propagation path from M to V as well as message-sending paths from C to V and V to M. A state change of M automatically updates V (Figure 3.2).

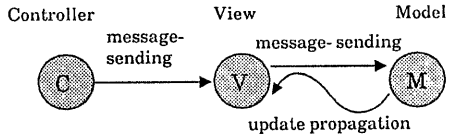

Figure 3.2 The internal structure of each pad.

Each pad has an observable part of its state. This part is represented as a list of slots. Each slot may be either a value or an attached procedure. The first slot of each pad is called the primary slot.

When a pad P_2 or its icon is pasted on another pad P_1, IntelligentPad constructs two paths between their views as shown in Figure 3.3; an update-propagation path from P_1 to P_2, and a message sending path from P_2 to P_1. This establishes the

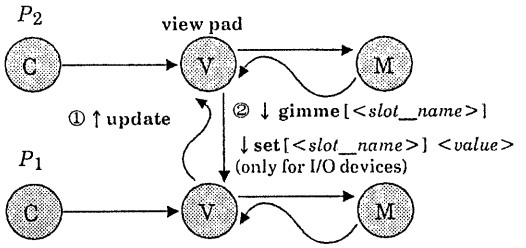

Figure 3.3 Standard message interface between pads.

connection between two pads, and defines a dependency from P_1 to P_2. The pad P_2 is a subpad of P_1, while P_1 is the master pad of P_2. No pad may have more than one master pad. If P_1 has more than one observable slot, we have to select one of them to associate it with P_2. This selection can be specified on a connection sheet pad associated with the connection.

Each subpad can send a **set** message 'set [<*slot_name*>] <*value*>' to its master pad. If the specified slot is a data slot, its value is updated with the sent value. Otherwise, the master pad executes the attached procedure. A master pad, when its state is changed, sends all of its subpads an **update** message without any parameter. Each subpad, when receiving an **update** message, sends back a **gimme** message 'gimme [<*slot_name*>]' to its master pad. If the specified slot is a data slot, this message reads out the slot value. Otherwise, the message makes the master pad execute the attached procedure. The returned value of the gimme message is set to the primary slot of the sender. Each procedural slot may have two

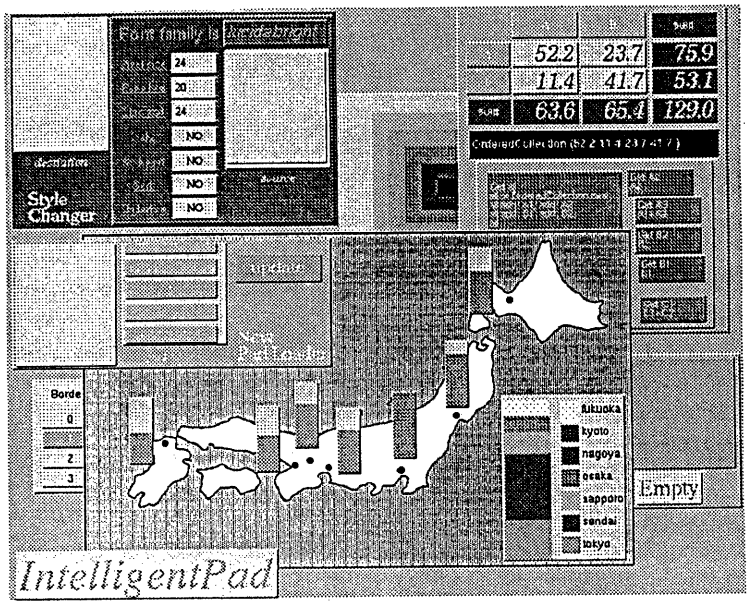

Figure 3.1 A display snapshot of the IntelligentPad system.

different attached procedures for the **set** and **gimme** messages. IntelligentPad allows us to disable some of these messages, which can be specified on the connection sheet pad. The two messages **set** *s v* and **gimme** *s* sent to a pad *P* are forwarded to its master pad if *P* does not have the slot *s*.

Besides the three standard messages, any pad can send some standard messages for geometrical operations to its master as well as to its slaves. Among them are move, copy, delete, hide, show, open, close, resize, and paste messages.

3.3 Shared State and Shared Workbench

IntelligentPad allows us to make a shared copy of any pad. A shared copy *P'* of a primitive pad *P* shares its model component with *P*, but has a dedicated view component and a dedicated controller component. Shared copies of a composite pad is defined to share the state of their base pads. IntelligentPad allows us to electronically mail a shared copy of an arbitrary pad to other users at different sites. Its receiver can share the state of this dynamic media object with its sender. IntelligentPad allows us to associate each shared copy with an arbitrary number as its priority level for the conflict resolution among concurrent user events.

While shared copies allow any state change operation at either site to cause the same change at the other site, they cannot make one site simulate paste operations performed at another site. This requires the event sharing, which a field pad can perform. We may put any pads on a field pad to define a shared workbench. A shared copy of this composite pad has a nonshared copy of each pad on a shared copy of the field pad. The primary slot of a field pad stores the last operation performed in the area defined by this pad. A field pad can detect any user-event request over itself and to convert this event request to the primary slot value of itself. The user events here includes mouse dragging, mouse clicking, and keyboard typing. A shared copy of a field pad receives an update propagation from the original, converts the new primary slot value to the original event request, and applies this event to the target pad on this copy. The field pad and the shared-copy mechanism provide fundamental facilities to build CSCW (Computer Supported Cooperative Work) systems.

In the IntelligentPad, each mouse event is detected by the controller of the top pad at the mouse location. Every user event over a field pad, however, needs to be sent to this field pad. This requirement implies that each pad, when put over a field pad, has to replace its controller with a special controller used only over a field pad. This controller called a field controller, when it detects a user-event request, sends its information to the view of the topmost field pad under this pad. The event information consists of the event type and the event location. The event location is given as the (x, y) coordinate relative to the field pad. The reason for the division of the display object for a pad into its view and its controller lies in this requirement to change its controller.

When its state is updated, the field pad traces up the subpad links to find the topmost pad at the given event location. This pad is the same pad that sent the event information. The field pad then sends the view of the target pad the same message as its original controller would directly send if it were not replaced. The same replay mechanism works in each shared copy of the same field pad, which realizes a shared workbench.

3.4 STAGE PAD AND SCRIPTS

In IntelligentPad systems, we can perform our jobs through synthesizing and manipulating various pads. Each pad when operated reacts to perform a subjob. In this sense, each pad may be compared to an actor who plays a role following the director's order. Its user is compared to the director. Some applications such as CAI and computer games need to program the sequence of operations on pads. Since everything is represented as a pad in IntelligentPad, this sequence of operations should also be represented as another pad. We call this pad a stage pad since it provides various actor pads with a stage on which they perform a play. The director of this play is hidden behind the stage pad as its script program.

Our script programs differ from the HyperCard script programs in the following respects. They do not directly send messages to actor pads to manipulate them. Instead, they send actor pads user events such as mouse clicks, mouse drags, and key strikes. They do not directly send messages to any actor pad to read or write any observable slot values. Instead, they read and write only the primary slot value of each actor pad. In other words, our script programs operate the actor pads in the same way as we operate them.

In an actual play, its script does not refer to any actor. It refers to his or her role. Any replacement of actors requires no rewriting of the script. The same is true to stage pad scripts. Each pad may have more than one operation point with different reactions. Hence, script programs refer to operation points instead of actor pads. Operation points are referred to through their role names. Each stage pad has a casting list, which is a table that associates operation points with their roles in its script program. The performance on a stage pad

allows interruptions by its user. Scripts are described as event-driven programs.

A stage pad with some actor pads on it can be pasted on top of another stage pad. This allows us to define a play within a play.

3.5 Pad Base

We need efficient mechanisms both for the management of pads and for the retrieval of specified pads. IntelligentPad provides three kinds of access methods: (1) a visual catalog of pads for browsing, (2) hypermedia networks for navigation, and (3) form bases and pad bases for quantified retrieval of pads. Any pads can be managed by these three search methods. This means that we can even store and retrieve CSCW environments composed with a field pad, CAI environments, or computer games, both composed with a stage pad, as well as documents, charts, tables, and ordinary deskwork tools.

Form bases cannot meet our needs to manage all types of pads in an integrated way. IntelligentPad provides a special folder pad called a pad base to meet such needs. A pad base is a database in which stored records are all pads. IntelligentPad may define any number of mutually exclusive pad bases. Pad retrieval from each pad base requires some method to quantify the pads to find. IntelligentPad has adopted the Query-By-Example approach for the quantification of the pads to retrieve.

Each pad has its nickname, registration number, composition structure, and the observable values of its constituent subpads. Each of these may appear in retrieval conditions. The registration number of a pad specifies its type. We will define the signature of a pad as follows. It is a tree with a pair of a registration number and a value at each node. The root node's pair is the registration number and the observable of the base pad. Each son of its root has a subtree representing the signature of a composite pad that is directly pasted on the base pad. The pads to retrieve can be quantified by partially specifying their signatures. Such a specification partially expresses a signature tree, leaving some of its nodes, subtrees, and/or the observable values of some nodes unspecified.

If we specify a computation pad pasted with a digital-display pad, a pad base will retrieve every composite pad with a digital-display pad pasted on a base computation pad. The retrieval result includes various hand calculators with different sizes and layouts. If we further specify the digital-display pad value to be 5.0 in the above query, the retrieval result excludes such a pad with its display value different from 5.0. For an insertion of a pad into a pad base, we only need to place the icon of this pad over the pad base icon.

The current version of IntelligentPad has a pad base pad that works as an interface to GemStone object-oriented DBMS. A pad base pad accepts a partially specified signature and returns search results (Figure 3.4). IntelligentPad also provides

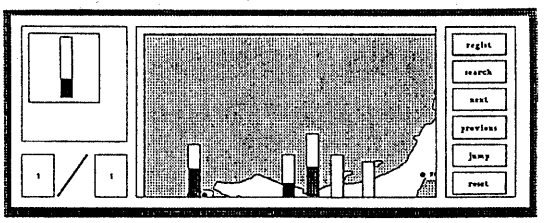

Figure 3.4 A pad base pad has retrieved a pad.

the signature-extraction pad. When a composite pad is put on this pad, it extracts the signature of this composite pad. If you paste this pad on a pad base pad, you may just put a computation pad with a digital-display pad as its subpad on the signature-extraction pad to specify the same query as above.

We finished with the first implementation of the pad base pad in 1990. Our experience with GemStone however has proved that its indexing technology can not sufficiently speed up the context search. The same problem lies in most OODBMSs available today. There are two reasons. They are based on the refinement-based programming paradigm, while the pad world is based on synthetic programming paradigm. In the former paradigm, complex object structures are defined in classes and their hierarchy. The indexing is applied to a set of objects in the same class. In the latter, however, complex object structures are defined by combinations of object instances. Each pad may have different structure from others. The IntelligentPad project developed a new indexing method to speed up the context search. Its details will be soon published elsewhere.

4. CONCLUSION

The IntelligentPad System as a synthetic media system is an open platform that provides an overall integrated environment for human activities. An appropriate mapping of any specific application to a single pad or a set of pads will assimilate this application into the environment of pads. Any external object that is controled by the underlying system also can be assimilated into this integrated environment by introducing a pad that works as its proxy in this environment. The IntelligentPad System is open to any application programs. They can be easily assimilated into an integrated environment. Once they are represented as pads, they can be easily combined with other functions

by pasting their pads. A pad base provides an integrated management of all kinds of pads.

Two special pads play important roles in this environment. A field pad and a shared copy operation provides shared workbenches for a group of users distributed at different sites. They can share their workbench in real time. Each operation at any site is shared by every site sharing the same workbench. A stage pad on the other hand can automate an arbitrary portion of human operations in this environment of pads. These two pads are also pads, and hence they can be arbitrarily combined with themselves as well as with other pads.

We are also conducting two new projects; one for allowing pads to operate concurrently, and the other for making the system more portable. The second project is a joint project with three major Japanese computer companies. It develops The IntelligentPad System in C++ using The X Window environment.

We are also developing applications of IntelligentPad. Figure 4.1 shows its application to a CAI system for classical mechanics. In this system, students can creatively play with various tools and parts to find out physical lows by themselves.

REFERENCES

[1] J. J. Gibson, *The Ecological Approach to Visual Perception*, Houghton Mifflin Comp. Boston, 1979.

[2] Y. Tanaka and T. Imataki, "IntelligentPad: A Hypermedia System allowing Functional Composition of Active Media Objects through Direct Manipulations", *Proc. of the IFIP 11th World Computer Congress*, San Francisco, Aug. 1989, pp. 541-546.

[3] D. Tsichritzis, "Object-Oriented Development for Open Systems," *Proc. IFIP Congress '89*, San Francisco, Aug. 1989, pp. 1033-1040.

[4] N. Naffah, "The Future Office Automation," *Proc. IFIP Congress '89*, San Francisco, Aug. 1989, pp.745-750.

[5] Y. Tanaka, "A Tool Kit System for the Synthesis and the Management of Active Media Objects", *Proc. of the 1st International Conference on Deductive and Object-Oriented Databases*, Kyoto, Dec. 1989.

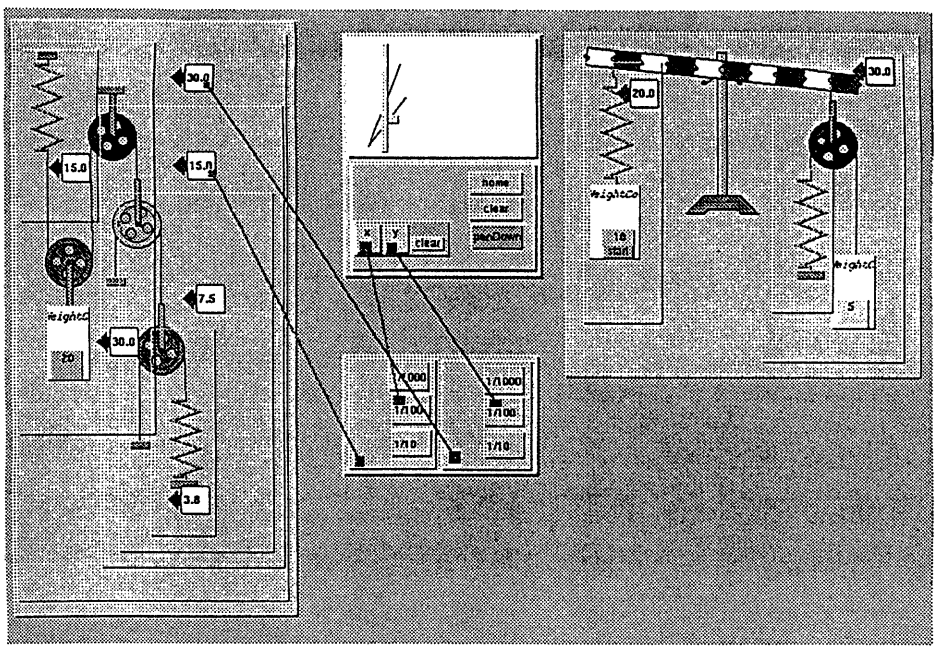

Figure 4.1 An application of IntelligentPad to CAI for classical mechanics.