An Object-Oriented Model for Interactive Multimedia Presentations

M. VAZIRGIANNIS AND C. MOURLAS

Department of Informatics, University of Athens, TYPA Buildings, Panepistimiopolis, 15771 Athens, Greece

In this paper a generic, platform independent design for interactive composite multimedia presentations is proposed. The design is concerned with temporal and spatial features of multimedia composition and synchronisation as well as handling the interaction with the user. A script based approach is adopted for modelling the presentation scenario and the user interaction. The overall design is based on the object-orientedparadigm.

Received October 1992

1. INTRODUCTION

Multimedia presentations are evolving as a popular and demanding field since the potential of new multimedia technologies is challenging a variety of groups of users. There is an apparent lack of tools to integrate the multimedia functionality and provide mechanisms for building interactive, complex multimedia presentations.

A resulting requirement arises for modelling of complex multimedia presentations that involve many multimedia objects transformed in various, media dependent, ways and presented according to a scenario or the user interaction.

Our effort aims at defining an object-oriented class that abstracts the composition and synchronisation features of interactive complex multimedia presentations and, moreover, provides the methods for the definition and execution of an interactive scenario. A prerequisite is a data model that supports specific attributes and functionality of the multimedia objects. Such a data model has been defined and partially implemented in our department [13]. Finally, platform independent design is another requirement that has to be fulfilled. The design presented herein is as generic as possible for platform independent solution.

The paper consists of three parts. The first part is a reference to previous related work done in our department. The previous work refers to modelling multimedia data [13] and extensions to a parallel language for supporting real-time applications [11]. Also we review important concepts of multimedia composition and synchronisation. There are sections referring to the actions that may be applied on multimedia objects (in space and time domains), the temporal relationships among actions and the synchronisation between them. Moreover, there is a presentation of how academic and commercial sites approach multimedia synchronisation and composition in time domain.

In the second part we define the composite item class, analyse and evaluate the design through an indicative example application. The model is defined in a notation similar to C++ and is based on previous pieces of work done in our department. The example refers to an interactive multimedia presentation of Crete. The example incorporates audio, video, text and image objects, buttons for handling user interaction and a script (scenario) that defines the spatial and temporal composition of the participating objects.

Finally, in the conclusion, we attempt to evaluate the proposed model in terms of the objectives initially set as well as in comparison to other systems referenced in the state of the art. Finally we propose future research directions.

2. STATE OF THE ART

In our department we have defined a data model that aims at representing the features and functionality of multimedia objects [13]. It is implemented as an object-oriented hierarchy, whose base class is called item. The item class abstracts mainly administrative features (like author, format, date of creation/modification etc.) of the multimedia objects. The media classes are subclasses of item. Each of the media classes inherits all the data and behaviour of the item class, while it defines the additional attributes and methods that are specific to each media type. Currently there are developed classes for text, audio, video and images. Classes for animation and music are in the specification phase.

Moreover in the department there is active research on language extensions to support the implementation of real-time (RT) applications [11]. The correctness of such systems depends not only on the logical results of the computations but also on the time at which the results are provided. In addition, the development of complex interactive multimedia applications presents similar characteristics and requirements with the RT applications since a desirable sequence and ordering of a set of actions in the time domain is required. For this reason much of our previous work in the field of RT

processing has been incorporated in the model that follows.

In the following there is an analysis of the several aspects of complex multimedia presentations. These aspects are:

- multimedia composition;
- actions and temporal relationships among them;
- multimedia synchronisation;
- multimedia temporal composition.

2.1. Multimedia composition

There are three aspects in the composition of multimedia objects [1]: spatial, temporal and configurational. Spatial composition refers to the spatial features of multimedia objects (i.e. video-animation overlay, video-text overlay etc.). The spatial relationship of objects are generally expressed by their location in space (a set of co-ordinates in space at which a given action may take place) while the temporal relationships are expressed in terms of time co-ordinates (a set of values in time where a set of actions may happen) for the objects [4].

Temporal composition refers to the temporal relationships between objects that participate in a complex presentation (synchronisation and temporal sequencing of components i.e. an audio starts at time $t0$, at time $t1$ a video starts and both fade out in the time interval $[t2, t3]$). Temporal composition includes also real-time behavioural properties of the objects since there is a need to define timing constraints for the execution of an action or define the maximum time interval for an object to wait during synchronisation. These properties will be discussed in later paragraphs where we propose a set of structures in order to describe these various real-time requirements of the objects. The motivation for temporal composition comes from the requirements for modelling situations where a number of multimedia components are simultaneously presented. Television and films are two obvious examples, each containing both audible and visual components. A more detailed discussion about the introduction of the time in programming systems as well as the problems of the verification and validation of such systems can be found in [9] and [10].

The configurational aspects refer to relationships which indicate the connections between the input and output port of components. The proposed model does not deal with these aspects because we aim towards a generic platform and device independent design.

An important concept in multimedia presentations is the scenario which may be viewed as a predefined spatial and temporal sequence of presentations of multimedia objects. It may either be static or dynamic (i.e. the flow of actions may be modified according to events that may happen).

2.2. Actions and temporal relationships among them

It is important to know and classify the actions that can be applied to multimedia objects. In [4] the actions are defined as arbitrary acts and are classified into atomic and composed ones. Atomic actions cannot be subdivided into partial actions for the purpose of synchronisation while composed actions consist of atomic or other composed ones whose parts have to be synchronised. This classification is made mainly for synchronisation purposes so that atomic actions are used for synchronisation scenaria. The start and end points of an action are used as synchronisation points. Little [2] classifies the actions performed on multimedia objects into unary (which adjust the multimedia objects according to the presentation requirements) and binary (which compose the adjusted objects according to the presentation script).

The topic of temporal relations between actions has been addressed by Allen [6]. In this paper there is a definition of a complete set of temporal relations between two actions. These are: *before*, *during*, *overlaps*, *starts*, *ends*, *equal* and the inverse ones (this does not apply to *equal*). Adding vacant time intervals the set is extended with the sequential, parallel first and parallel last relations.

In [4] there is a set of path operators that are based on the aforementioned relationships. This set defines the semantics of a synchronisation mechanism and the synchronisation of the presentation of multimedia objects. The most important operators which are also incorporated in the proposed model are: $\wedge$, $\vee$, *. They are defined in [4] as follows:

- $A \wedge B$ (parallel-last): Actions A and B are started at a common start point and are executed concurrently. The composed action terminates when all the participating actions (A and B) terminate.
- $A \vee B$ (parallel-first): Actions A and B are started at a common start point and are executed concurrently. The composed action terminates when the first in time participating action (either A or B) terminates.
- $A^{i}*$ (repetition): Action A will be repeated i times.

2.3. Multimedia synchronisation

Synchronisation in the context of multimedia refers to the mechanisms used by processes to coordinate the ordering in time domain. According to [1] the objectives of multimedia synchronisation are: starting and stopping multimedia objects at desired time points, establishing or removing connections between components at transition points and ensuring global synchronisation between activated components.

The main concepts used in [1] to achieve synchronisation are the world_time and object_time that are necessary in defining a low level synchronisation scheme. The world_time concept as it is defined in [1] is used as a reference time and refers to the time elapsed from the origin of time, which is usually the start of the current application. The object_time concept refers to a multimedia object internal time. Assuming that the composite C comprises the multimedia objects Ci then the criterion

for C to be synchronised is that for all its components Ci the following formula holds [1]:

C.world_time − Ci.world_time < dt

where dt is the synchronisation tolerance. Synchronisation then is maintained using temporal transformations and jumps.

In general, the synchronisation of multimedia objects presentation has two aspects:

- *low level synchronisation*, which addresses the problems of hardware capability for preserving certain temporal features of multimedia composition within a range. For instance the synchronisation tolerance referred in [1] is such a feature.
- *high level synchronisation*, which refers to the sequence of actions on multimedia objects and handling of events that occur and cause other actions.

The proposed model does not deal with the low level synchronisation since the objective is the definition of a generic, abstract design for modelling the functionality of multimedia composition and synchronisation. Moreover, it is assumed that the underlying hardware and software is capable of proper execution and presentation of the composed multimedia objects.

2.4. Multimedia temporal composition

There are various approaches in temporal composition of multimedia objects. According to [14] temporal composition of multimedia specifies how actions (performance activities) are organised and combined in the time domain. The temporal composition is abstracted in an object-oriented model including three class hierarchies. The first one serves scheduling purposes (*ClockScheduler* class), the second triggers actions (*Trigger* Class) and the third one deals with the events caused from the beginning and the end of an action (*BeginEvent* and *EndEvent*). Also temporal transformations for all time dependent media are provided.

Another effort for modelling composite multimedia is the evolving standard MHEG (Multimedia and Hypermedia information coding Expert Group) [12]. As for temporal composition MHEG defines structures that support three types of synchronisation:

- *elementary synchronisation*, which refers to two objects or actions ($O1, O2$) and the respective time intervals ($T1, T2$). The elementary synchronisation is classified into sequential ($O1$ is executed $T1$ time units after the application start, $O2$ is executed $T2$ time units after the end of $O1$) and parallel ($O1$ is executed $T1$ time units after the application start while $O2$ is executed $T2$ time units after the application start).
- *chained synchronisation*, in which there is a set of objects ($O1, O2, \ldots$) and the respective time intervals ($T1, T2, \ldots$). For instance $O1$ is executed $T1$ time units after the application start, $O2$ is executed $T2$ time units after the end of $O1$, $O3$ is executed $T3$ time units after the end of $O2$ and so on.
- *cyclic synchronisation*, in which a set of events is repeated at specific time intervals.

In the commercial field MACROMIND DIRECTOR is one of the leading tools. MACROMIND DIRECTOR is a product that runs on APPLE Macintosh platforms and is used for the preparation of interactive multimedia applications. It internally uses the LINGO language [7] which, among other features, supports temporal composition of media objects. Temporal features of LINGO deal with timer manipulation (there are system timers and a general timer that, using the timeOut feature, generate events). LINGO also supports event manipulation. There are certain event types: keyDown, mouseDown, mouseUp that are recognised as events and invoke the so-called event scripts or macros. Also the timeout event which is caused when one of the aforementioned timers reaches the timeout point set by the system or the user. For instance the command

when ⟨event⟩ then ⟨macro | script⟩

invokes a script or a macro (both are pieces of LINGO source code) when the specified ⟨event⟩ occurs.

3. METHODOLOGY AND COMPOSITE ITEM CLASS DESCRIPTION

We aimed at the definition of an object-oriented framework for multimedia composite modelling [3]. An important construct of the proposed design is the action concept, its definition and classification. As actions we accept acts which aim to manipulate spatial or temporal features of one or more multimedia objects. Actions are classified in unary and n-ary actions. Unary actions apply to single multimedia objects. They are mostly methods or combinations of methods from the media classes referring to spatial or temporal presentation features. Examples of unary actions are: image scaling, cropping, filtering, colouring, text formatting, audio cueing, playing, suspending, resuming, stopping, inverting and temporal scaling. A n-ary action applies to a set of multimedia objects and defines a way of group manipulation of the set referring to spatial or temporal space. Such actions are spatial overlaying, overlapping, grouping, ungrouping, moving and temporal synchronisation of presented multimedia objects.

In the proposed model three temporal operators are used in order to construct complex (n-ary) actions using elementary (unary) ones. The temporal expressions generated in this way define complex actions which include the notion of synchronisation between the elementary ones. The three temporal operators which have been already defined are: $\wedge$ (parallel-last), $\vee$ (parallel-first) and * (repetition) operators.

Another important concept in the proposed model is the notion of the events. Events are arbitrary distributed points in time used for the ordering of the actions and their coordinated access to shared resources. A similar approach for synchronisation of multimedia systems has been proposed in [5]. Events may be regarded as a special kind of information passing mechanism between objects, since the information transmitted is only the start and end of a specific action. More precisely, within the lifetime of every object there are two special events which are generated:

- the start point of a specific action;
- the end of that action.

If two or more media objects are combined in a multimedia presentation the above events will be used in order to assure the desirable temporal order of presentation. For example, if an event which has been generated at the end of a specific action is used to trigger the execution of another action, then we have the 'sequential' synchronisation. If the event which is generated at the start of an action is used to trigger the execution of another action, then we have a 'parallel' synchronisation. More complex synchronisation conditions can be formed if more media objects are combined for a multimedia presentation.

Moreover, selecting a specific button, during one presentation, the corresponding event is caused. In this way, as it will be described in the following paragraphs, asynchronous interaction with the user can also be handled by the model. Also two special events timeout and no_sync can be produced in case of a time constraint violation or from synchronisation problems.

The handling of interrupts is also supported by the model. When we refer to interrupts we mean the asynchronous control signals that can be sent to the application by pressing special control keys (e.g. ^C) or by a hardware failure.

As already mentioned, the proposed design does not involve low level synchronisation issues regarding the capability of the hardware for proper presentation of data intensive multimedia information. For the proposed design, synchronisation refers to the level of sequence of actions and handling of events.

A composite item class is defined for modelling the data and behaviour of complex multimedia presentations. A composite multimedia item is an entity that embeds a set of multimedia objects along with their spatial and temporal space presentation features. It is clear that the data are independent from the presentation. In the composite object there are only references to the locations of the actual data, and the actions that will be applied to these data during the presentation. The multimedia objects are not altered themselves and may be shared among various applications. The class description covers requirements for interactive as well as static scenarios. A partial description of the composite_item class written in C++ follows:

```
composite_item {
//administrative data
char                *author;
date                creation_date;
attr_value          properties<(attribute,value)>;
composite_item      father;

...

//spatial data
obj_list    graphics_ob_list(<id,pos,scale_f>);
obj_list    text_ob_list(<id,pos>);
obj_list    image_ob_list(<id,pos,scale_f>);
obj_list    button_list (<id,pos>);
obj_list    composite_list(<composite_id>);

//temporal data
obj_list    audio_ob_list(<id,invert,t_scale_f,t_s,t_d>);

//spatio-temporal data
obj_list    video_ob_list(<id,invert,pos,scale_f,t_scale_f,t_s,t_d>);

//temporal composition
tuple_list    scenario_list<start_time, duration, <action>, synch_events, exception_handler, interrupt_handler>;

//methods                                                                    //

play();
stop();
pause();
self_synchronise();
add_scenario_tuple (start_time, duration, <action>, synch_events, exception_handler, interrupt_handler);

...
};
```

The composite_item class includes administrative data regarding author, creation date, and the properties list. This list (which is author defined) consists of a set of attribute value pairs, where attribute is a semantic feature of the composite and value is the local value of the attribute. For instance a property value pair would be: (political event, elections of 1990). There is also the identification of the father composite which invokes the current composite_item and returns control to it once its execution has finished. The top level composite has null value for the father variable.

Multimedia object composition (spatial and temporal) may involve extensive usage of unary actions (intermediate transformations) in order to have the multimedia objects prepared for *n*-ary actions. These intermediate transformations require a lot of processing power especially in the case of distributed multimedia data [2]. The spatial and temporal data lists included in the class description serve the purpose of preparing the data for composition (i.e. unary actions like scaling, cropping, cueing etc.) during an initialisation phase and not during presentation time. Again it has to be stressed that the actual multimedia data are not altered since they are independent of the application.

For each media type there is a list of the objects that participate in the composite. The lists include the identification (id) of the objects along with variables which refer to both spatial and temporal features of the participating objects. The identifications (id) of the objects do not locate the actual data but the location of the transformed data according to the spatial, temporal and spatio-temporal lists. These are the ids that are used in the ⟨action⟩ attribute of the scenario_list tuples.

The composite_item class offers attributes for modelling the presentation features of the participating objects in space and time. The spatial attributes refer to the location of the object in the viewport (pos) and the spatial scaling factor (scale_f).

The temporal attributes refer at first to the temporal scaling factor (t_scale_f) which indicates the scaling transformations in time for time depending media (video, audio, animation etc.). Also they refer to the direction of presentation which depends on the value of the boolean variable invert. If it is true (T) then the object presentation direction is inverted otherwise it is normal. Finally they refer to the temporal portion of the object that participates in the composite. This feature is defined by the start time (t_s) and duration (t_d).

The user interaction, which is an essential part of a multimedia presentation, is mainly served by buttons. The composite_item class includes the button_list variable which is a list of buttons that are included in the composite. The button structure is assumed to be provided by another class that models the button generic data and functionality. The rest of the variables refer to the spatial and temporal composition of the composite's parts. The presentations may be static (i.e. the sequence of presented items is predefined) or interactive (i.e. the flow of actions is affected by events that may happen). The composite_list variable is a list containing the identifiers of the composites that are invoked from the current composite item.

The most important part of an interactive multimedia application is the scenario or script concept. It defines the temporal sequence of presentation for the objects as well as the handling of various events exceptions and interrupts. The scenario concept is supported in the composite_item class by the scenario_list variable which is a list of user defined tuples. Many of the attributes of each tuple have been introduced in programming languages for the implementation of real-time applications [8, 11]. Each tuple of the scenario list has the form:

⟨start_time, duration, ⟨action⟩, synch_events, exception_handler, interrupt_handler⟩

where

- start_time determines the start of execution of the actions described in the tuple. It may be of the form:

  (Abs_time and Event_tree) or
  (Abs_time or Event_tree)

  where the starting time may depend either on absolute time (Abs_time) or on sequence of events (Event_tree) or on a combination of both. The sequence of events is a general logical expression of events. In this context, events are used for the ordering of actions in time domain. As already mentioned events are caused either by the start or the end of an action or by a user interaction event (e.g. selection of a button). An event may cause the execution of a unary or *n*-ary action. For instance, when we need a task to start at 12:45 or at the moment that either event1 or event2 has happened the value of start_time will be (12:45 or (event1 or event2)).
- Duration determines the duration of the action in the tuple. It may be expressed in absolute time units or as a set of events that will cause the execution of the tuple to end. Similarly to start_time it has the form:

  (Abs_time and Event_tree) or
  (Abs_time or Event_tree)

- ⟨Action⟩ is the list of actions that will be performed and may refer either to an individual object (in this case actions are multimedia classes method calls e.g. play, move, display) or to a combination of objects (like overlay, overlap, resolve occlusion, scaling, cropping etc.). Moreover we can construct complex actions using elementary ones and the temporal operators defined earlier. These expressions of actions include the notion of synchronisation between the actions implicitly resulting to a more elegant design of the whole presentation. In addition, with these temporal expressions we reduce also the number of events used for explicit synchronisation with the result

of a simple description of a complex multimedia application. For example, the composed action (video.play ∧ audio.play) implies that the video and the audio will start at the same time and be executed concurrently. The composite action will be terminated when both the two participating actions will be terminated.

- Synch_events refers to the naming of the two special events which are generated at the start and the end point of the describing ⟨action⟩. It is an ordered pair of event names where the first name represents the event that will be occurred at the beginning of the execution of the defined ⟨action⟩ and the other is the name of the event that will be occurred at the end point of the action. These names of the events defined in this field are used for the explicit synchronisation of the actions participating in a scenario. More precisely, we use these names in the expressions start_time and duration as we have already seen in the description of these fields above. For example, the pair (e1,e2) for the composed action (video.play ∧ audio.play) means that when the start_time expression is evaluated to true then the event e1 will be generated. At the end of this composed action the event e2 will be also generated. If we do not care about the generation of an event we place an underscore (_) instead of its name.
- exception_handler is a set of actions that will be executed in case of violation of the predefined time constraints or when there exists a lack of synchronisation between the participating actions. As a result, the execution of the specific action will be suspended and the exception_handler will take immediately priority. This action will handle the exception and will try to bring the whole system to a desirable state. One of the two events timeout or no_synch will have occurred showing that the exception produced from a timing constraint violation or from synchronisation problems of the action respectively.
- interrupt_handler is an action that has to be performed immediately after a system failure or by typing special control characters (e.g. ^C). Such failures can be produced for instance from the hardware, resource failures, environmental factors etc. and they are application independent. When a failure occurs the system receives an identifier of the interrupt which is usually a small integer and then calls this interrupt handling routine. After the execution of the handler the execution resumes at the point where it was interrupted.

Using the proposed model, we can easily describe multimedia applications that do not interact at all with the user as well as applications that the user can play a significant role in the whole presentation. In the former case the sequence of the actions and the ordering of these actions in the time domain is predefined. Consequently the model is used in this case in its deterministic and sequential form.

However, the power of the proposed model is demonstrated in the development of complex interactive multimedia presentations. In such presentations the asynchronous way that the user interacts with the system have to be handled in order to satisfy the desirable sequence and ordering of the actions. This asynchronous interaction is handled by the model supporting the notion of the buttons and the interrupt handling facility. The selection of a button generates a corresponding event which in its turn can cause for example the suspension of the current presentation and the concurrent execution of others.

Moreover, the user can interfere in the presentation sequence sending special control signal (e.g. by pressing ^C) which can be handled by the model through the support of interrupt handler definition. As a consequence, the proposed model seems suitable enough for the design of simple as well as complex and interactive multimedia applications.

3.1. A sample composite multimedia presentation

We assume that there is the need for an interactive multimedia presentation concerning the Greek island Crete. The required multimedia material is available in a multimedia data base under the aforementioned multimedia data model representation scheme. The presentation refers to historical, tourist and physical information of Crete and consists of several composites composing a hierarchy. The composites HI1, PI1, TI1 are invoked from compl, while TI2, TI3, TI4 from TI1. The presentation layout of each component may be seen in Figure 1. The contents of the compl composite object is following:

```
compl {
text_ob_list ("CRETE", 2, 10);
graphics_ob_list (⟨crete_map, 80, 40, 0.8⟩);
audio_ob_list (⟨cretan_mus1,F,1,0,10⟩);
button_list (⟨Archaeological, (50, 5)⟩, ⟨Touristic info, (50, 125)⟩, ⟨Physical info, (50, 250)⟩, ⟨Close, (300, 100)⟩);
composite_list (⟨HI1⟩,⟨TI1⟩,⟨PI1⟩);

scenario_list
(
⟨0 or e2,close.button_up,cretan_mus1.play(),(_,e2),,
  on ^C cretan_mus1.stop()⟩,
⟨3,close.button_up, crete_map.display(dissolve),,,⟩,
⟨Archaeological_button.button_up,,HI1.execute(),,,⟩,
```

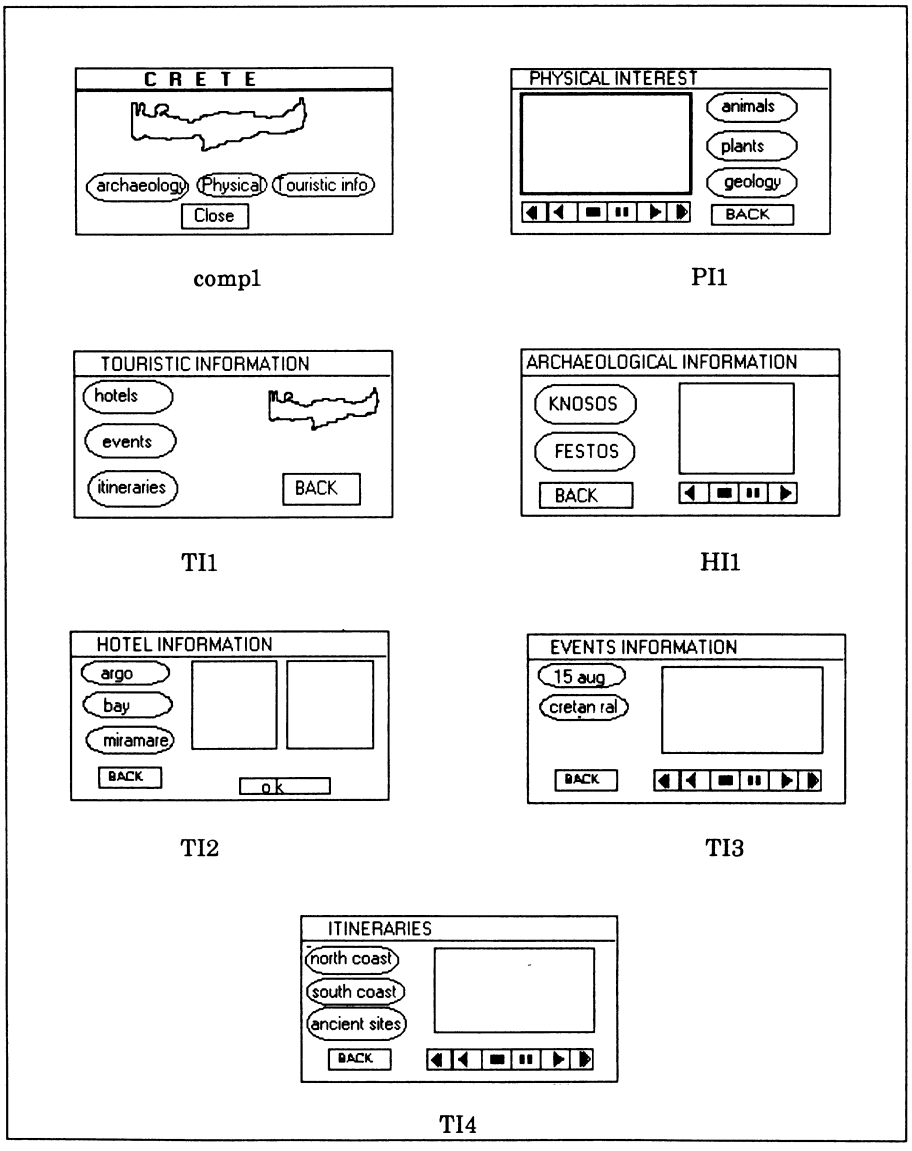

FIGURE 1. The presentation layout of the composites participating in the example.

```
⟨Touristic_info_button.button_up,,TI1.execute(),,,⟩,
⟨Physical info_button.button_up,,PI1.execute(),,,⟩,
⟨close.button_up,,exit,,,⟩
);
}
```

An analytical description of the composite compl follows. It includes the CRETE text resource that will be displayed at the position (2, 10) relatively to the composite viewport origin (the same applies for the position of all objects participating in the composite). The graphics_ob_list includes the map of Crete as graphical object at the position (80, 40) scaled by the factor 0.8. The audio_ob_list includes the first 10 seconds of cretan_mus1 audio object that will be played at normal speed (t_scale_f = 1). The composite list indicates that the composites HI1, PI1, TI1 are invoked from within the compl composite. There are four buttons (Archaeological, Touristic info, Physical info, Close) located at the respective positions: (50, 5), (50, 125), (50, 250), (300, 100). The scenario list consists of six tuples. The first tuple indicates that the cretan_mus1 audio will be played starting from the origin of the application (0) or when event e2 occurs. Namely the cretan_mus1 is played back repeatedly. The music execution will last until the close button or if ^C key are pressed. In the former case composite's execution is stopped while in the latter only the music execution stops. The second tuple indicates that the crete_map will be displayed using the dissolve effect at the third second of the composite's execution. The following three tuples are the invocation of the three composite items HI1, PI1, TI1. The last tuple indicates that if the close button is pressed the execution of the whole composite stops. The contents of the PI1 composite item is following:

```
Pl1{
button_list (
⟨animals, 20, 200⟩,⟨plants, 80, 200⟩,⟨geology,140,200⟩,
⟨rewind,340,40⟩,⟨stop,340,120⟩,⟨play,340,160⟩,⟨pause,340,200⟩,
⟨play_invert,340,200⟩,⟨fast_forward,340,280⟩,⟨BACK,340,350⟩);

audio_ob_list (
⟨animals_voice, F, 1, 0, 23⟩,
⟨plants_voice, F, 1, 0, 14⟩,
⟨geology_voice, F, 1, 0, 32⟩,
⟨mus_back, F, 1, 0, 30⟩);

video_ob_list (
⟨animals_vid, F,(20,30),1,1,0,14⟩,
⟨plants_vid, F,(20,30),1,1,0,19⟩,
⟨geology_vid,F,(20,30),1,1,0,34⟩);

// the actions that will take place on Pl1 invocation.
scenario_list (
⟨0, back.button_up, Pl1.display(),,,on^C Pl1.stop()⟩,

// tuple referring to activation of animals button
⟨animals.button_up,                          //start condition
 stop.button_up or e4,                       // duration
 ( (animals_vid.play() ∨ animals_voice.play())
    ∧
   (plants.disable() ∧ geology,disable()) ),    // actions to be taken
 (_,e3),                  // end of the composite action will cause e3
  self_synchronise(), // exception handler in case of synchronisation problem

on ^P generate_event(e4) // interrupt handler causes event e4
                         // and stops the execution of the composite
                         // action

⟩,

// On e3 event the plants and geology buttons will be enabled
⟨e3,,(plants.enable() ∧ geology.enable()),,,⟩,
...
//the two following tuples indicate the functionality of the pause button
⟨
  pause.button_up,          //start condition
  pause.button_down,        // duration
  &video.pause(),           // the VCR pauses
  ,,,
⟩,

⟨pause.button_down,,&video.resume() ∧ all_buttons.active(),,,⟩,

// The functionality of the BACK button
⟨BACK.button_up,,exit,,,⟩,
...,
}
```

4. CONCLUSION

Composite multimedia is a quite complex problem because of the various issues related to spatial and temporal features of multimedia data. An effort for composite multimedia modelling should be abstract enough so as to represent the functionality and the data required for a generic platform independent model but also specific enough and technically complete so that this model may be instantiated on various implementations. The proposed model is an initial effort towards multimedia composition modelling. The attractive points of the design are:

- *static and interactive scenarios.* They describe the spatio-temporal features of a multimedia presentation. From the reviewed approaches only MACROMIND DIRECTOR supports the scripts concept. The other systems embed the functionality as methods in structures that are used.
- *platform independence.* The proposed model is designed so as to be platform independent since the functionality included is generic and widely accepted. It is assumed that there are tools in the various platforms that are able to implement the aforementioned functionality.

- *user defined events.* The proposed design enables the author define names for various events: start and end points of a unary or *n*-ary action as well as for asynchronous situations (user interaction). These events may be used for synchronisation purposes. MACROMIND DIRECTOR 7 and de Mey [14] also support events but not defined by the user.
- *spatio-temporal composition.* The model provides the spatial, temporal and spatio-temporal lists which include the identification of the objects that participate in the presentation along with their transformations in space and time. From the reviewed models and tools only the MHEG[12] and de Mey [14] provide spatial and temporal transformations of the participating objects.
- *interrupt and exception handling.* None of the reviewed designs provides the definition of exception and interrupt handlers as the proposed model does.

This design may be applied in a variety of application fields where interactive multimedia presentations are desired. Such applications may be educational, commercial advertisements, information points etc. A long term objective is the definition and implementation of a language for authoring complex multimedia presentations based on the model that has been described. Since in multimedia presentations there is the need for concurrent execution of media objects, we intend to design the aforementioned language including parallelism features.

REFERENCES

[1] S. Gibbs, L. Dami and D. Tsichritzis, An object-oriented framework for multimedia composition and synchronisation, *Object Composition*, Centre Universitaire d'Informatique (Univertitaite de Geneve), pp. 133–143 (1991).
[2] T. Little and A. Ghafoor, Spatio-temporal composition of multimedia, *IEEE Computer*, **24**(10), pp. 42–50 (1991).
[3] B. Cox, *An Evolutionary Approach to Object Oriented Programming*, Addison-Wesley (1989).
[4] Petra Hoepner, Synchronizing the presentation of multimedia objects, *ACM SIGOIS*, pp. 19–31 (1991).
[5] R. Steinmetz, Synchronisation properties in multimedia systems, *IEEE Journal on Selected Areas in Communications*, **8**(3), pp. 401–412 (1990).
[6] J. F. Allen, Maintaining knowledge about temporal intervals, *Communications of the ACM*, **26**(11), pp. 832–843 (1983).
[7] Macromind director, *Interactivity Manual*, Macromind Inc (1990).
[8] Y. Ishikawa, H. Tokuda and C. W. Mercer, Object-oriented real-time language design: constructs for timing constraints, *Technical Report CMU-CS-90-111*, Carnegie Mellon (1990).
[9] Shem-Tov Levi and Ashok K. Agrawala, *Real-Time System Design*, McGraw-Hill (1990).
[10] J. A. Stankovic, Misconceptions about real-time computing—A serious problem for next-generation systems, *Computer*, **21**(10), pp. 10–19 (1988).
[11] C. Mourlas and C. Halatsis, Extensions to a parallel prolog system to support real-time applications, *In Proceedings of the 4th International PARLE Conference*, pp. 551–565 (1992).
[12] ISO-IEC, Multimedia and hypermedia information coding experts group (MHEG), *JTC1/SC29/WG12*, pp. 70–72 (1991).
[13] M. Vazirgiannis, M. Hatzopoulos and I. Rizos, HADT: Hypermedia application development tool for tourist applications, to appear in *European Journal of Information Systems*, MacMillan Press. 1993.
[14] V. de Mey, C. Breitener, L. Dami, S. Gibbs and D. Tsichritzis, Visual composition and multimedia, *Object Frameworks*, Centre Universitaire d'Informatique (Universite de Geneve), pp. 243–258 (1992).