Multimedia phase-spaces

PETER BØGH ANDERSEN. DEPARTMENT OF INFORMATION AND MEDIA SCIENCE, UNIVERSITY OF AARHUS

To appear in: *Multimedia Tools And Applications,* Kluwer Academic Publishers. German translation in: Martin Warnke, Wolfgang Coy, & Georg-Christoph Tholen (eds.): *Hyperkult. Geschichte, Theorie und Kontext digitaler Medien.* Stroemfeld/Nexus: Basel, 1997, 191 - 226.

1. What are interactive multimedia?

In this introductory section, I shall discuss the nature of interactive multimedia, and their present mode of development.

1.1. Multimedia are documents, music or film

When new media appear they are normally treated as new versions of old media (Madsen 1994): the film posed as theatre, television as filmed radio, and video as electronic film. This is also true of interactive multimedia which are often conceived of as a book or collection of documents with more flexible ways of turning pages (Apple's Hypercard, Allegiant's Supercard, Netscape), as a film or piece of music that can be stopped and started (Macromind Director, Apple's Quicktime) or as a new kind of theatre (Laurel 1991).

However, all applications are computer systems, and reveal themselves as such at closer inspection, so why this playing hide-and-seek with the user? Why pose as a book when one is really an object-oriented program?

The reason is the paradox inherent in creating a new medium. On the one hand a medium is only a medium if it is used for communication on a social scale. This means that an audience must be willing to buy the products. But the audience will only do that if it get satisfaction for its money. A prerequisite for this is that a contract is established between audience and authors specifying conventions of interpretation and use of the products. But such contracts are normally learned by the audience by being exposed to products as a part of its socialization, which again presupposes that medium and

contracts already exist. This vicious circle — medium -> audience -> contract -> medium — can only be broken by unfolding it in time: at each point of time, a suitable contract must exist which enables the consummation of the products and which is slightly perturbed for each new product.

Thus, the birth of a new medium with new genres must take time, since at each point of time it must survive and change.

What is, then, the new feature that sets interactive multimedia apart from other media? Most authors agree that interactivity or non-linearity — the possibility for the user to physically influence the product during consumption — is an important characteristic of the medium (see e.g. the papers in Landow 1994). I would add that if interactivity is not exploited, it would be much better to use the older media, since the quality of computer text, pictures or film is much lower than that of the corresponding traditional medium.

Whereas the film or the book denotes one discourse, the interactivity of the computer system means that it can denote a (possibly infinite) set of discourses. It can denote a *world* in which many kinds of stories may unfold. In fact, the discourses of a computer system only exist as potentials (Aarseth 1994).

A main problem in designing interactive products is that two parties are struggling for power: the user wanting to influence the story, and the author insisting on his ideas. One step towards a solution is to view the computer as an *elastic* medium. It means that the user's physical actions on the medium constitute its characteristic nature. The user can physically change the product during the use process, and this very change is the essential ingredient of the aesthetics of computer systems.

But the computer medium is not merely a raw material. Like the author, the systems designer must impose an aesthetic form onto the computer, but the aesthetic qualities of this form consists in the way it yields to or resists the user's physical actions on keyboard or mouse. The aesthetic form lies in the *elasticity* of the system.

1.2. The Eye of Wodan.

The system used as illustration in this paper is just another small step leading from the old media to the new one.

Instead of seeing the interactive multimedium as movies and documents, it proposes to see it as a dynamic phase-space. The notion of a phase-space is not used as a metaphor, but constitute the basic building block of the implementation of the system. Section 6 gives some illustrative examples.

The system is called *The Eye of Wodan.* It is a multimedia system about the Danish Viking age designed for the museum *The Vikings of Ribe* (See Fig. 1.1. Ribe is a small town in the south-east of Jutland)

Fig. 1.1. The museum.

The system uses graphics, movies, music, and speech. Interaction takes place via a large touch-screen (Fig. 1.2), and design of system and room is integrated.

Fig. 1.2. The touch screen.

Fig. 1.3. The ceiling contains 31 bulbs.

The system controls the lightning of the room (Fig. 1.3) and projection of silhouette pictures on the walls (Fig. 1.4).

Fig. 1.4. The silhouette pictures of one of the walls.

Project members were: Lars Andersen, technical programming. Peter Bøgh Andersen, direction and programming. Helle Juhl Andersson, research and manus. Berit Holmqvist, acting and sound. Bjørn Laursen, graphics and room design. Steffen Zacher, light.

The ideas in this paper have been developed over several years, mostly in connection with design of concrete systems (see Andersen & Holmqvist 1990, Andersen 1992, 1995a). However, the Eye of Wodan system is the largest one, and I shall use that as the main source of examples.

Not all the theoretical ideas were actually used in the Eye of Wodan, and sometimes they were implemented differently than described in this paper, the work method being to test ideas concretely first in a quick and dirty fashion, then make abstractions and generalizations. I mostly present the *a posteriori* generalizations, but if they have resulted in major re-conceptualizations, I also describe the actual running implementation.

2. Multimedium are dynamic phase-spaces

The notion of phase-spaces comes from physics where it is used to describe the global behaviour of a large collection of smaller objects, such as molecules. For example, the behaviour of water can be described as a phase-space with two dimensions, temperature and pressure. The space is divided into three parts, solid state (ice), fluid state (water) and gaseous state (steam). However, the present system is not primarily intended for physical simulation, but rather as a tool for creating interactive aesthetic products.

2.1. Phase-spaces and trajectories

Imagine you are Alice in Wonderland and you come across a glass cube with a fly inside. You stop and watch its curious behaviour. You discover that when the fly approaches you, its buzzing grows louder, and when it retreats the sound diminishes.

Fig. 2. 1. Object: A fly.

Furthermore, when the fly soars, it grows larger, when it descends it shrinks, and, finally, when it flies to the right it gains visibility, whereas it becomes invisible as it moves to the left. See Fig. 2.2.

What Alice experiences is a *phase-space* of the fly with three dimensions, loudness, size and visibility. The fly itself is represented as a point in the space specified by a vector of three components, <loudness, size, and visibility> whose values are typically numbers. If they range from 0 to 255, the vector <5, 10, 255> represents a nearly silent, small, and completely visible fly (the lower right fly in Fig. 2.2).

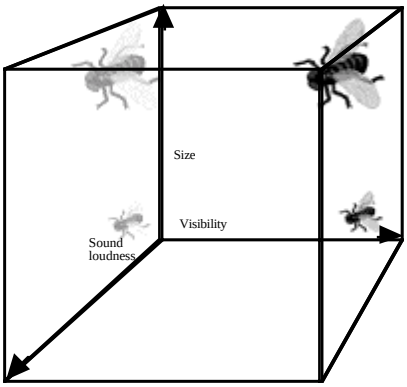

Fig. 2.2. Points of phase-space differing in size and visibility.

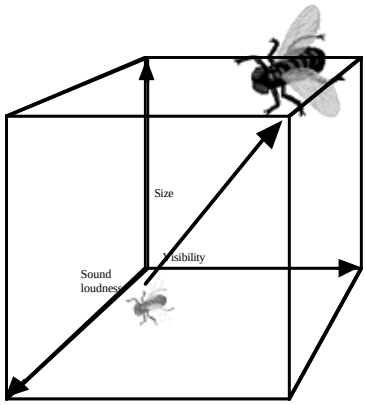

Fig. 2.3. Trajectory of phase-space: fly grows larger, more visible and noisier.

Changes of the object are represented by *trajectories* in the phase-space. For example, a line from <5,5,5> (nearly silent, small, and invisible) to <255, 255, 255> (buzzing loudly, large, visible) represents an event where the fly grows larger, more visible, and more noisy (see Fig. 2.3). This is naturally interpreted as the fly approaching Alice (see Fig. 2.4).

Fig. 2.4. The state-changes generated by the trajectory of Fig. 2.3. Possible interpretation: fly approaches.

2.2. Manifestation

Apart from the phase-space itself and a set of trajectories, we also need to specify how the dimensions of the space are *manifested* as visible or audible signals. In the fly-example, the loudness dimension is manifested as sound, whereas the size and visibility dimensions are manifested as properties of the picture representing the fly. In a real implementation, the values of the sound-

dimension would be input to the sound-manager of the machine, the size-values determines the horizontal and vertical dimensions of the bitmap picture, and the visibility dimension controls the RGB components of the pixels of the bitmap.

In our example, the phase-space dimensions and the manifestation dimensions can be made nearly *continuous*, so that the changes can be smooth. We can calculate the values of the manifestation from the values of the phase-space dimensions. The same is true of locations. Given the location of an object living in a 3-dimensional space plus the position of the eye, we can calculate how the object shall be manifested on a 2-dimensional screen.

However, in some cases the mapping is not continuous, either because of the nature of the manifestation, or because of limitations of the machine. In this case we speak of *discontinuous manifestation.*

The former is the case when the phase-space is manifested by phenomena that by nature are discrete, such as e.g. verbal language. Suppose that the fly does not merely buzz but is able to talk (after all, we are in Wonderland). Suppose furthermore, that the greetings of the fly depend upon its mood and its knowledge of Alice. Now, even if for other reasons we want to distinguish 256 moods of the fly, we cannot map the continuous scale of moods into a continuous scale of greetings. We have to divide the space into subspaces that all are manifested by the same sentence (Fig. 2.5.).

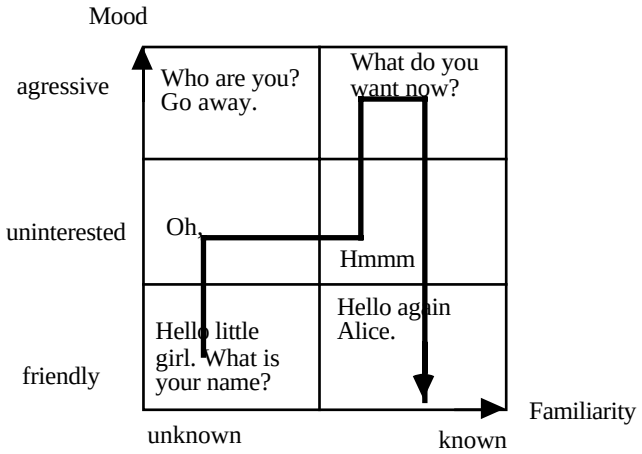

Fig. 2.5. Discontinuous manifestation. Verbal language.

The trajectory in Fig. 2.5 represents a story where the fly starts as friendly and unacquainted with Alice. It soon grows uninterested, as Alice starts being familiar. Then the fly becomes aggressive, but eventually ends as friendly. The convention is that as long as the trajectory stays within a subspace, the same sentence is used. Only if it transgresses the border between two subspaces does the speech change.

The problem is relevant in e.g. learning applications, where verbal responses should depend upon the knowledge of the reader accumulated so far.

In The Eye of Wodan, effect sounds were produced in this way. As we shall see later, the system contains large pictures over which the user can move his gaze. The location of his gaze is represented by the *iris* of an eye, as shown in Fig. 2.6 where the user is looking at the merchant in the stern of a Viking ship. The iris can be moved over the ship and represents the part of the picture that is visible to the user. I shall sometimes call it a *view*.

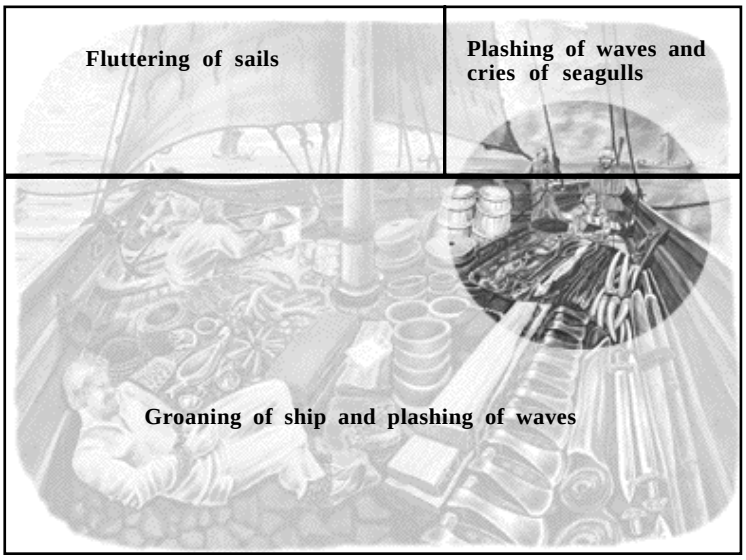

Fig. 2.6. Sound carpet of the merchant's ship.

The effect-sounds are manifestations of this iris living in the two-dimensional world of the background picture. In Fig. 2.6 the sound part of the iris is manifested as fluttering of sails in the top left part of the picture, as waves and sea-gulls in the top right part, and as waves and creakings of the ship in the lower part. Since the location of the iris (its midpoint) is located in the lower part, its sound manifestation is "groaning of ship and splashing of waves". The background effect-sounds change as the user moves his eye — a trick also used in film editing.

Other cases arise because of the limitations of the hardware. At present, only few machines can produce interactive 3-D graphics that are aesthetically satisfying. Either the modelling has too low resolution so that the cubes, triangles and spheres are all too visible, or the changes are so slow that the jumps bother the eye.

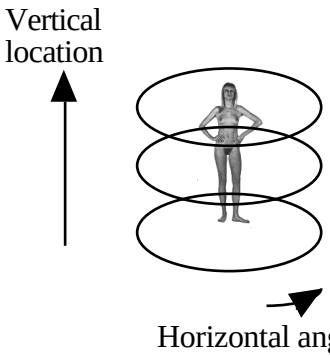

Fig. 2.7. Sampling discrete manifestations.

In such cases selected pictures can be produced off-line, and assigned to areas in the phase-space.

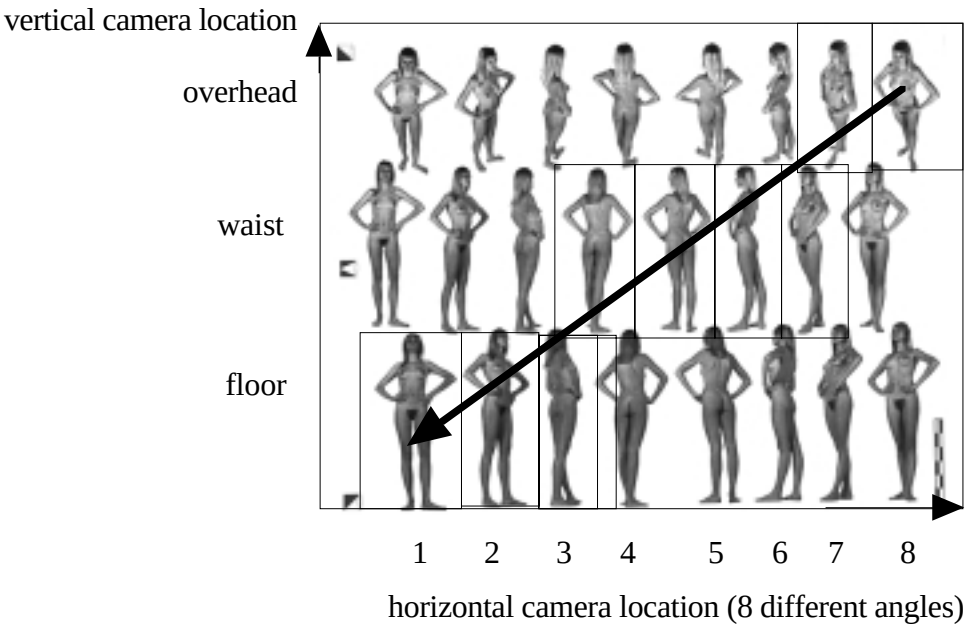

Fig. 2.8. Discontinuous phase-space of female body. Dimensions: vertical and horizontal camera angles. A trajectory is shown where camera descends and rotates simultaneously.

Fig. 2.8. shows an example. It is borrowed from *Illustrator's Figure Reference Manual*, a source-book for artists. Each model is represented by 24 pictures, 8 pictures for 3 levels of camera positions, each level containing 8 pictures with a distance of 30º (Fig. 2.7). As shown in Fig. 2.8, the resulting phase-space of the camera has two dimensions, vertical and horizontal camera location. A similar technique is used in Apple's QTVR Player.

As before, camera movements are represented by trajectories in this discontinuous space. The manifestation convention is also the same: as long as the trajectory stays within a subspace, the picture indexed by the coordinates of the subspace is shown on the screen.

The separation of phase-spaces and manifestations enables us to specify movements in the phase-space without committing us to the way it is realised. As mentioned above, if we have a fast machine, we can build a 3D model of the body and let the trajectories move the model which is subsequently drawn on the screen. However, if the hardware is slower, we can index a collection of ready-made pictures with sub-spaces of the phase-space as shown in Fig. 2.8. In the first case we get continuous movement, in the latter the movement is discrete, one picture replacing the other, possibly mediated by a fading operation.

3. Using the room as manifestation

Fig. 3.1. The Eye of Wodan. Eye, iris, and pupil. Manipulated photo: Bjørn Laursen.

Fig. 3.2. The monk Adso, alter ego of the user. Photo: Bjørn Laursen.

Manifestation of phase-spaces should not be limited to the computer-screen. In systems that aim at giving the audience an experience, the whole room should be seen as a possible locus of manifestation. This was in fact done in the Eye of Wodan project where the multimedia room was designed by Bjørn Laursen to support a global experience. Fig. 3.1 shows the large touch-screen displaying the system. The screen is an eye — in fact the eye of Wodan, the highest god in Nordic mythology. Through this eye the user can get a glimpse of the bygone age of the Vikings.

Fig. 3.3. The shoemaker.

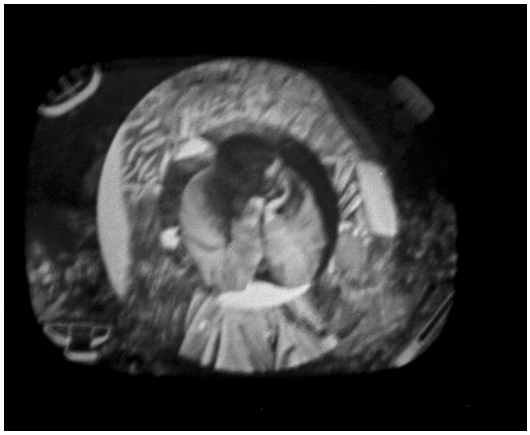

Fig. 3.4. The shoemaker's wife.

To the left is Adso, the user's alter ego. Adso is an English monk visiting Ribe in order to gather information for the Christian missionaries (see also Fig. 3.2). In Fig. 3.3 - 3.4. the eye is open, displaying an iris (the large circle), and in Fig. 3.4 a pupil (the round picture in the middle) has been added.

Adso is as ignorant of the heathen Viking culture as the audience, and can thus ask questions about the subjects the user may want to know about, but being a contemporary, he does not spoil the illusion.

Fig. 3.5. The silhouette wall. Memory is vivid.

If the user is inactive, the eye closes, and only a menu is left (Fig. 3.7). Simultaneously the light is dimmed to a cold blue, and sounds of rain begin. The voice of Adso also changes: from being a healthy young man, his voice becomes an old man's voice. In addition, the pictures of the silhouette wall (Fig. 3.5) disappears, and only a lonely moon remains (Fig. 3.6).

The idea is that all we see is really Adso's memories. Adso is an old man who tries to remember his youth. The present is dismal and cold; only when the audience uses the screen and activates the dormant memories of youth, is a trace of the fullness of youth recreated for a brief moment: rain gives way

for larks, the noise of artisans, or music; the light becomes warm yellow; silhouettes of houses and people replace the cold moon on the silhouette wall.

Fig. 3.6. The silhouette wall. Memory vanes. Photo: Bjørn Laursen.

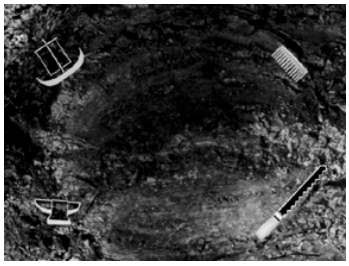

Fig. 3.7. The closed eye. Photo: Bjørn Laursen.

This mode of interaction was designed to let the audience themselves experience an important norm of the Viking culture: only men (and at that time it was really only men) that displayed courage, curiosity, and cleverness could achieve the fullness of life. Dull people and cowards were *níðingr*. *Níðingr* were not evil people but people that lacked honour and richness, and therefore would never go to Valhalla after death. Whereas Christianity — and Adso — saw life as full of negative/positive dichotomies (good/ bad, sinners/saints, man/God) the Vikings measured life on a scale of fullness ranging from 0 and upwards (Carlsen 1994).

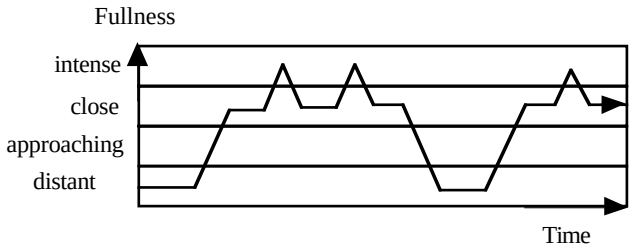

Fig. 3.8. Trajectory of the room: oscillating between intensity and distance.

Instead of verbalizing this information, the system lets the audience experience it: if they act as *níðingr*, they are punished by the closing of the eye, the disappearance of silhouettes, and the emergence of cold blue light.

This global change of atmosphere is controlled by the one-dimensional phase-space shown in Fig. 3.8. It has four values representing Adso's distance from his youth: *distant, approaching, close* and *intense*.

As indicated by the trajectory in Fig. 3.8, the system oscillates between these four values. The values are of course manifested on the screen by the

opening eye, but also in the whole room in terms of sound, light and wall-pictures (see Table 3.1.)

Fullness	Sound	Light	Silhouettes
intense	speech + birds etc.	red	mythological masks
close	birds, music, sounds of artisans, etc.	yellow	close ups of persons
approaching	speech + rain	blue + yellow	pictures of distant village
distant	rain	blue	no pictures except moon

Table 3.1. Manifestation of the phase-space of the room. The Fullness dimension.

4. Dynamic phase-spaces

In the preceding sections I have shown several examples of trajectories. The next question is: how are these trajectories created? A brilliant paper by Ogawa,, Harada, & Kameko 1990 presents one solution. They describe a multimedia system consisting of three-dimensional objects; two dimensions are the screen-coordinates and the third one is location in time. Thus, all objects have a time dimension. The dynamics of the system is described by means of time-triggers, that generate and delete objects as time passes. Another variant is developed in Bøgh Andersen 1992, 1995a. This is the one discussed in this paper.

4.1. Force fields

In Section 1.1 I argued that interactive multimedia must be elastic, but how can we technically create an elastic medium? It is not a good idea to design a set of trajectories of the kind we saw in the previous section. The reason is that a particular trajectory only describes what happens to an object that is located on the trajectory, but has nothing to say about objects outside it. So either the system is inactive most of the time, or we have to force the user into the trajectory. And once caught inside the trajectory, we cannot allow the user to escape, since the action of the system outside the trajectory is undefined. We will end up with a system that presents a limited set of options to the user; when the user has chosen an option, the system takes control and stays in control, e.g. showing a film or a slideshow, until the trajectory is finished. Then the steps are repeated.

A better possibility is to define a *force-field* on the phase-spaces that specifies a trajectory for all points in the space. In this solution, the next step

of the system is defined for all states of the object, and we can let the user act much more freely.

Here is a simple example. We define a potential z by means of the equation

$$(1) \qquad z = \frac{h}{dist(x+a, y+b)^2 + 1} \text{ where } dist(x,y) = \sqrt{x^2 + y^2}$$

and use the gradient of the potential to control the velocity of the system. *h* is a constant giving the basic force of the field, and *a, b* are displacements of the field in relation to the origin. Fig. 4.1 shows an example where the basic force is -5 and *a* and *b* are zero. The result is a "hole" in the energy landscape. If an object is placed at the border of the hole, it will roll down to the bottom where it will find rest, since the gradient there is 0. The ball in Fig. 4.2 illustrates the point.

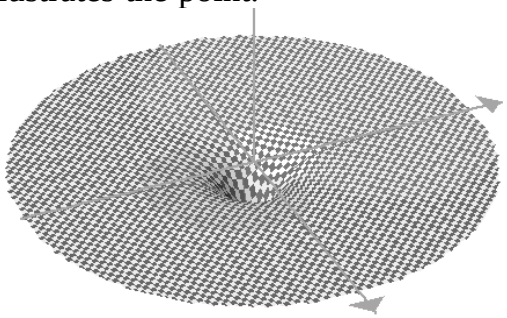

Fig. 4.1. Two-dimensional dynamic phase space.

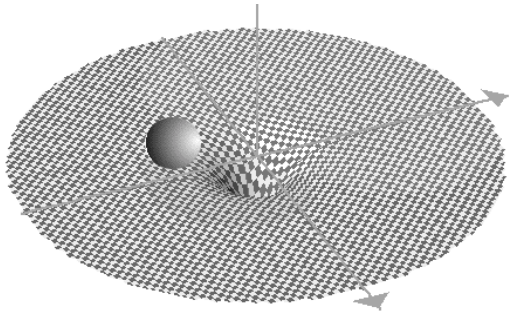

Fig. 4.2. As 4.1. The ball represents the system.

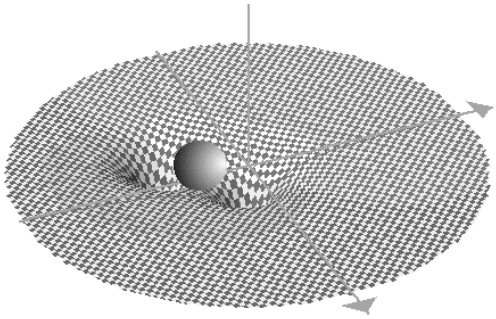

Fig. 4.3. The user has added another potential. The total potential is the sum of the parts.

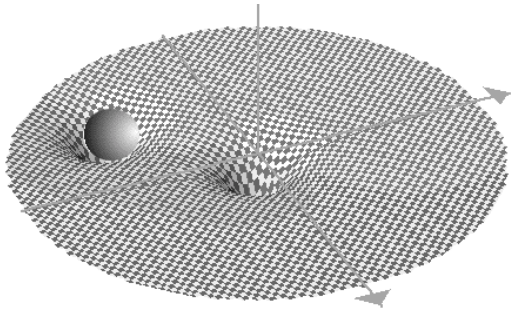

Fig. 4.4. The user drags the ball by moving his potential.

We can prevent the ball from falling by placing another force-field in the vicinity (Fig. 4.3). Now the ball rests uneasily on top of a ridge, and a small perturbation can make it slide down in the new hole. If the new hole is moved away (by changing the *a,b* parameters from 3,3 to e.g. 5,5) then the ball follows (Fig. 4.4).

Mathematically, such holes are called *attractors*, because they attract systems in their vicinity. Our hole is only a *local* attractor, since it looses its power as we move away (assuming that the system has some inertia, so that a minimum gradient is required to set the system in motion). Martin Warnke has kindly pointed out to me that potentials reminiscent of (1) are used in natural science too, but here the gradient represents acceleration, not velocity. This observation underlines the basic aesthetic use of mathematics of this paper: I am primarily interested in designing movements that signify something and the choice and usage of formalisms is an aesthetic issue. Use of the gravity equation in the "proper" way is motivated if we choose a naturalistic aesthetic.

4.2. Perturbation and Compensation

This method is inspired by Thom's catastrophe theory (Thom 1983, 1990). Thom too uses phase-spaces with a field of force defined on them. The fields are defined by means of potential functions such as the one-dimensional cusp

(2) $x^4 + ax^2 + bx$

x is called the *internal* variable and a,b the *control* variables. The force field can be made to change by changing the control-variables as illustrated in Fig. 4.1 - 4.4 where a new attractor appears and begins to move.

In the system I have used Thom's general ideas of internal and control spaces, where the control space is said to *perturb* the internal space.

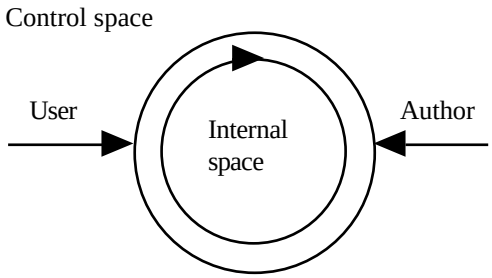

Fig. 4.6. User and author as perturbators of force-field.

In this framework two main processes are taking place in the multimedia system. In the internal space, the system is moving along the gradient field of equation (1). We call this process *compensation*; its iterative nature is illustrated by the closed loop inside the system in Fig. 4.6. The forcefield can be changed from the outside by the user and the author. As in Thom's theory, the change is effected by changing parameters of the equations defining the

force field which I call *perturbation*. The basic process in the system consists of perturbations followed by compensations.

This architecture has the following advantages

1. *Flexibility:* since the development of each state of the system is defined, it is not necessary to confine and keep the user in a small subset of system states. Exploring can be real. This presupposes of course that the author invents force-fields that have aesthetic or pedagogical meaning in all locations of the phase-space, which is certainly a non-trivial task (see Wildgen 1982, 1985).
2. *Robustness:* since the development of the system is well-defined everywhere, there is hope that systems break-downs in unforeseen situations may become rare.
3. *Autonomy:* the system possesses an autonomy of its own, since neither author nor user may manipulate system objects directly. This autonomy could become the locus of the system's inner aesthetic form, which author as well as user must subject themselves to (Aarseth 1995).
4. *Conflict support.* Effects of conflicts can be represented in a systematical way, namely by summation of its component gradients. This is very useful because all interesting narratives must have some type of conflict in order to maintain the interest of the reader. In addition, the conflict between author and reader can be represented as well.
5. *Unpredictability.* Realistic systems will be partly unpredictable, maybe even have chaotic phases. This means that the author is not able to foresee the possible readings of the system. The author creates a world with objects and forces which the user can view as a *real* object of exploration.

In the following I shall describe how the author can influence the system.

5. Narratives: author perturbations of the phase-space

The main tool for the author to influence the phase-space is *events* organised in *narratives*. We shall only look at events.

Like interface objects, events are also objects located in a phase-space, but besides the properties of the phase-space, they contain three components, namely a *Trigger*, a *Failure* part and a *Success* part. All events work concurrently according to the rule in Code 5.1.

```
If Trigger is fulfilled then
 Perform Success part
else
 Perform Failure part
```

Code 5.1.

From Code 5.1, which is the only method of performing events, we can see that *only local constraints on events* are possible. The system does not contain any representation of a global narrative structure, so global structures, if any exist, are *emergent properties of the set of concurrent events.*

Fig. 5.1. shows the main input and output of events. The *Trigger* measures the hand of the user (e.g. the mouse or the touchscreen), the screen (e.g. which objets are visible?), and other events (values of properties of the events, i.e. their location in a phase-space). The *Failure* and *Success* parts influence the eye and ear of the user and other events (*eye*: e.g. scroll the screen, show a picture or a movie, change the lights of the room; *ear*: play a speak, an effectsound or some music; other *event*: move it in its phase-space).

The main dimension of events is *Modality*. This dimension represents the urgency or the degree of existence of the event. It is divided into three main sections, namely *Impossible*, *Possible*, and *Actual* (cf. Bremond 1966, 1970. Brandt 1989. On the notion of modality in programming, see Bøgh Andersen 1995a).

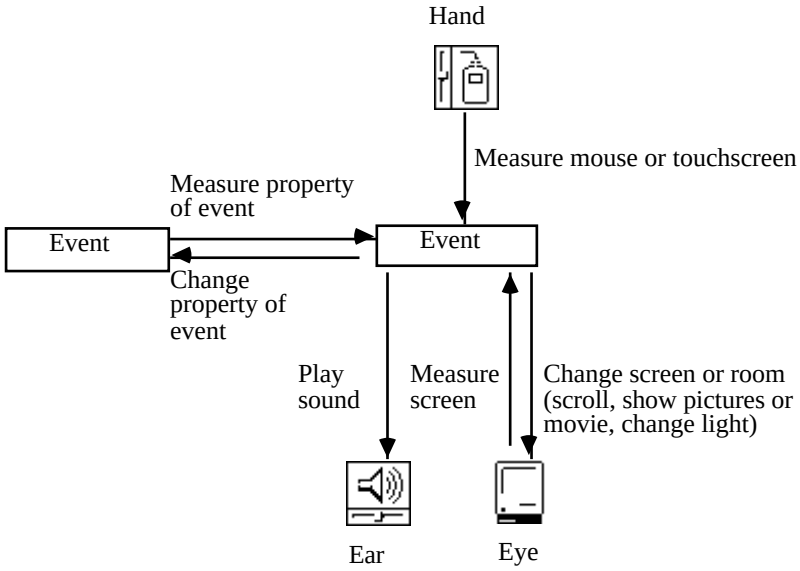

Fig. 5.1. Input and output of events.

The idea of a unique dimension of Modality was not used in the actual system, and the rhetorical figures described below were created by a combination of other methods. The notion was first used by three of my students in an interactive Winnie-the-Pooh system (Andersen, Johannsen, Mikkelsen &

Sams, to appear); since it simplifies and generalises the implementation, I shall use it in the presentation here (see Section 6 for the actual implementation).

When an event is located in the *Impossible* section, it cannot be selected to be performed; when it is located in the *Possible* section, it can be selected for performance; and when it has been selected and is performing, it is located in the *Actual* section.

Thus, the simplest life of an event is depicted in Fig. 5.2. In the first part of its life, it is *possible*, but is not elected for performance, because its trigger is not fulfilled. Then the trigger succeeds, the event moves into the Actual part, and stays there while is does its work. Finally, it stops and moves down into the realm of impossibility, being dead for the rest of the time.

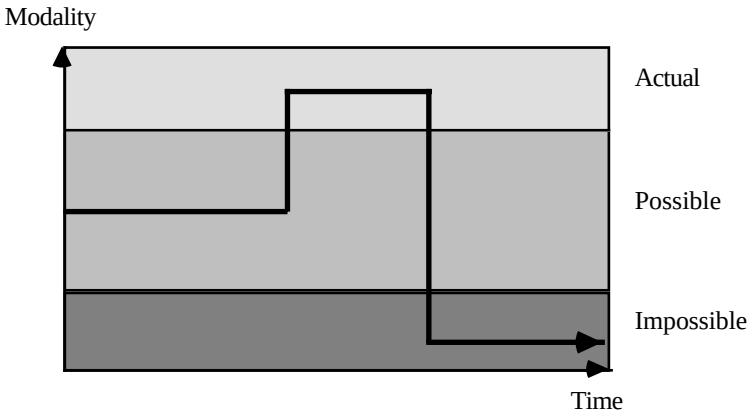

Fig. 5.2. The life of a simple event.

Other events may lead different lives. For example, some events may be repeatable, so they will not move to the Elysium of dead events, but return to the section of possibility, becoming eligible once more.

The exact location within the three sections makes sense in the context of the main loop that searches the possible events from the top in descending order to find one whose trigger will fire. Since the search starts from the highest modality, one can achieve an ordering by assigning different values of modalities to the events. When the highest ordered event A has been performed and demoted to impossible, the next higher B is eligible, etc. Thus, A will always precede B in the performance. Since we can change the modality in run-time, the ordering of events is dynamic. We shall see some uses of this facility in the next section.

5.1. Generalizations, associations, and descriptions.

Interactive systems must be designed by means of rhetorical figures in order to make sense. In the following I describe a collection of patterns that were used in Eye of Wodan.

Generalization and *Concretization* are very frequent text patterns. In the former case, concrete examples are followed by a general summary; in the latter case, a general statement is followed by concrete examples.

Generalizations were used in the module about trade (see Fig. 2.6). In the beginning Adso is an arrogant young academic who views the Vikings as ignorant rowdies. However, from the concrete information about the merchandise in the ship he comes to believe that the Vikings were knowledgeable and peaceful peasants and pedlars, and he expresses his changing attitude in terms of summarising generalizations. His attitude changes gradually, the expressions of respect growing stronger as time passes.

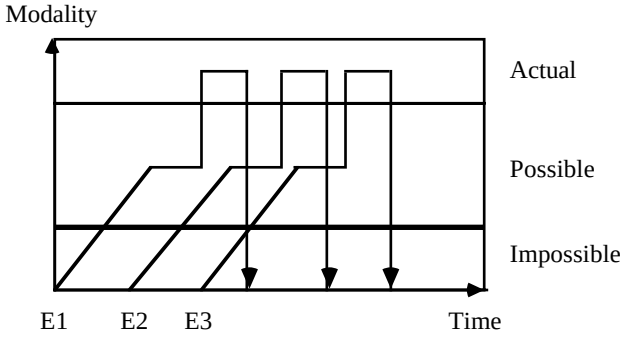

Fig. 5.3. Generalization

In Fig. 5.3, E1 - E3 express increasing respect for the Vikings. Inspection of the figure shows that all are impossible in the beginning, but events exemplifying knowledge on the behalf of the Vikings increase the modality of all three events. The increase is depicted by the rising lines. Eventually E1 moves into the Possible section, and when there is room for it is performed. Notice, that when E1 is performed, E2 and E3 are still in the realm of impossibility, so they are not eligible for playing. As new concrete information appears, E2 becomes possible and is performed, and so on for the rest of the generalizations.

The same concrete example can support more than one generalization. For example the information

(5.1)

Adso	Where do those whetstones come from?
Merch.	They are from Norway, I sell a lot of them here in Denmark.

shows that the Vikings sailed to Norway (increase *knowledge*) and that they traded heavy goods (increase *tradevolume*). The next information only increases knowledge since it shows that the Vikings had been in France:

(5.2)

Adso Those double-edged swords, I suppose you don't make them yourselves [...]

Merch. Those swords are made in this country, but most of them are in fact imported from the Frankish Empire [...]

(5.3) exemplifies a generalising remark on the Vikings' knowledge:

(5.3)

Adso You certainly bring commodities from far away, Norway, Sweden, Germany. Maybe you did know where England was?

Merch. Well, yes, now that I think of it Adso, I guess I knew all along.

This change of attitude is built into a *Counterpoint* composition. As can be seen in Fig. 2.6, a slave is lying in the bottom of the ship, and the more Adso expresses admiration for the Vikings, the more the slave cries for help. In the beginning he whispers, but his voice grows stronger, and in the end Adso's view is drawn down to him and the slave tells his story which reveals a quite different picture of the Viking culture: it was also a culture based on a slave economy.

The counterpoint composition builds up to a point of no return and a climax: the more Adso is reassured of the peacefulness of the Vikings, there more the suppressed truth reveals itself.

The counterpoint of the slave is also constructed as shown in Fig. 5.3. Only this time, the events are promoted by events, such as (5.3), that ascribe knowledge and peacefulness to the Vikings.

Generalising events like (5.3) are *free* in the sense that they have no trigger; they occur when they become possible and there is room for them. Events such as (5.1) and (5.2) are *bound* since they concern a particular object — whetstones and swords — and are therefore triggered by measurements of the screen: the object must be visible to the user (Fig. 5.4) and/or touched by the user (Fig. 5.5).

Fig. 5.4. Icon for view-sensitive trigger

Fig. 5.5. Icon for hand-sensitive trigger

Fig. 5.6. Icon for time-sensitive trigger

However, it is sometimes useful to insert a minimal pause between free events, since one runs the risk that they are played immediately after each other if

the user happens to be inactive. In this case we need a trigger that measures the time elapsed after the preceding event has stopped (Fig. 5.6). This turned out to be necessary in a similar rhetorical figure which we can call *Association*.

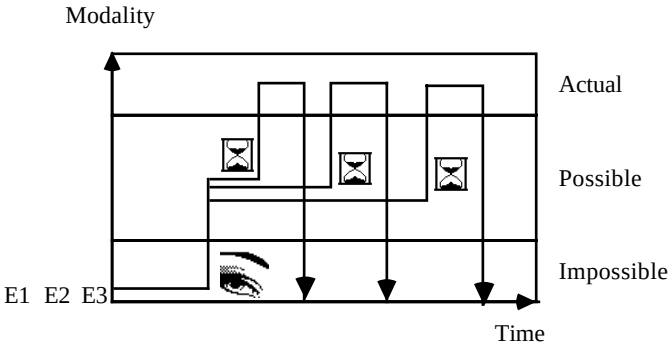

Fig. 5.7. Association

Association is different from Generalization in that one event (the trigger event) makes all events possible at the same time. The idea is that viewing a particular object activates a chain of associations in a person who verbalises the next association whenever there is room for it in the conversation.

For example, the smith in the ship is very fond of alcohol, and when the merchant mentions the wine barrels, wine, beer and mead occupy his thoughts for quite a while. Whenever processing time is available, he utters (5.5) and similar sentences.

(5.5)

Smith	Well, Adso, do you manage to drink some mead at your monastery.
Adso	No, we drink wine. But William of Baskerville - he is our abbot - has forbidden us to drink more than two and a half litres per day.
Smith	I suddenly feel a vehement desire to convert to white Christ.

It turned out that the Association became a sequential monologue because the user was not quick enough to select other subjects. This was prevented by using triggers that would only fire 30 seconds or so after the previous event had occurred.

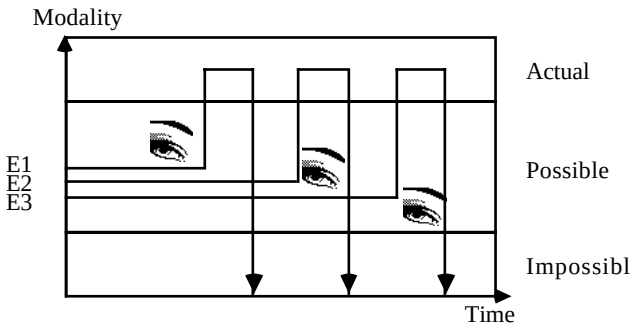

Fig. 5.8. Simple sequence.

A very frequently used — but also somewhat boring — pattern is the *Description* which gives information about particular objects, such as axes, cloth, swords, etc. In the system most descriptions consisted of more than one piece of information. The first one was presented the first time the user touched the object, the next one the next time, etc., so that the user could get a deeper knowledge of the object.

Descriptions follow the pattern of *Simple Sequence* in Fig. 5.8. When the module opens, all events are possible, each event having higher Modality than its successor. Each event has the same trigger, namely that the object has become visible and/or has been touched by the user. The effect is that the user gets a new event in the sequence each time he returns to or touches the object.

5.2. Interactive rhetorics: Forks and conversations.

Since Adso changes his mind during the narrative, it was often necessary to have two or three versions of the same event: one where he was still the arrogant student, and one where he displays a more humble attitude. The former was used when the "knowledge"-parameter (cf. above) was low, the latter when it had passed a critical limit. (5.6) illustrates the arrogant attitude in the case of a walrus-tooth.

(5.6)

Adso	Do you know what this is merchant?
Merch.	Do you Adso?
Adso	Yes, they are ivory teeth from an Olifant. It is a terrible animal living in a place called India. God has created many wonderful animals you never saw merchant. It is in the books at my monastery. You know, the griffin, Phoenix, and the unicorn.
Merch.	You don't say, well, that sounds exciting. But these happen to be walrus teeth from Norway.
Adso	Are you sure?
Merch.	Very. I can tell you that much of the ivory you have seen is really walrus-tooth.

In these cases, we have a *fork*, namely two options of which only one should be realised.

The fork was implemented by collective suicide, i.e. by letting each of the two event kill the other one when it died. The trigger of the humble version required that the force of Knowledge is larger than 1, whereas the conceited (5.6) fires when the force is smaller than 2. The effect is as shown in Fig. 5.9: only one of the events can be performed, the choice depending upon the value of *Knowledge*.

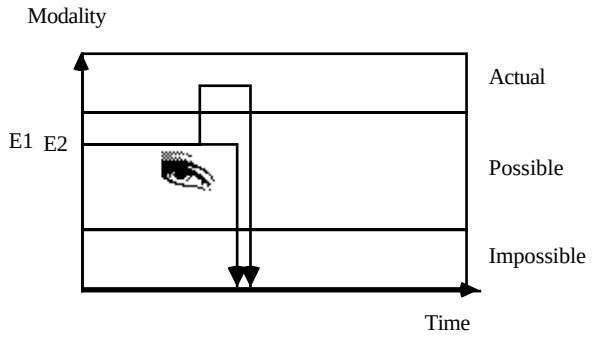

Fig. 5.9. A fork.

Other narratives were based on *conversations* between a group of Vikings. In the next example we witness a family quarrel. Ketill is aggressive and accuses his brothers Bjarke and Sigge of being men without honour and interest in fighting and looting.

In such conversations we need to distinguish between immediate replies to other events, and the story characterising the person himself. Since the replies must come as quickly as possible, they must have higher priority than the person's own story. In the speech *Accusation* Ketill accuses his brothers of being men without honour, and if Bjarke subsequently is chosen, he angrily replies by rejecting the accusation in the speech *Rejection* (See Example 1 below).

Accusation:
Ketill(angry): Mother says you are all coal-biters [...]
Bjarke(firm): She does not, I'm sure.
Ketill(breathless). Yes she does, you coal-biter ....[rambles on]
Rejection:
Bjarke(furious): If you were not a minor, you had called me coal-biter for the last time [...].

Example 1.

However, the rejection should occur only if the accusation is in the past.

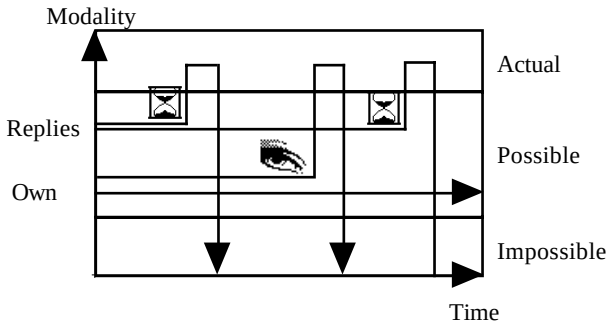

Fig. 5.10. Replies and own story.

Fig. 5.10 implements this by means of priority and an tense-trigger: the reply *Rejection* has priority 2, which is the highest one in that narrative, and it has a trigger that fires only when we see Bjarke and the *Accusation* in narrative *Ketill* is in the past. Bjarke's own story has priority 3 and fires whenever we see Bjarke.

The time concept used in the replies was implemented in a quick and dirty manner and a more systematic solution would be to add time to the modality-space of events, as we have in fact already done in all diagrams in this section. In this construction, an event becomes a point in a two-dimensional phase-space with dimensions Modality and Time. A time-modality memory could possibly be useful for dramaturgical planning of the events.

6. Implementation

The "Eye of Wodan" system is a particular instantiation of a more general object-oriented environment for designing multimedia systems based on the notion of dynamic phase-spaces. The environment was programmed in Metrowerks' CodeWarrior C++ programming environment for the Macintosh. In addition, editors for narratives and objects were implemented in Allegiant's Supercard. The editors allow the designer to work at a higher level than C++, and are able to generate C-code. This section describes illustrative parts of the system.

The system is built on a theatre metaphor (Laurel 1991) and its basic part-whole hierarchy is shown in Fig. 6.1.

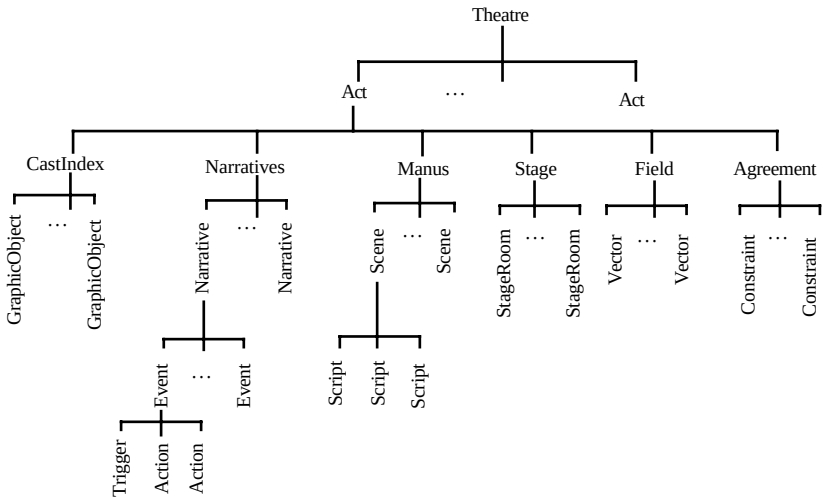

Fig. 6.1. Part-whole hierarchy of the most important classes.

The application itself belongs to class *Theatre* which consists of one or more *Acts* which consists of *Scenes* which again consists of three *Scripts*: the *BeginningScript* that generates the material used in the individual scene, i.e.

the *GraphicalObjects*, the *Narratives*, the *Vectors*, the background sounds, the movies, and the pictures; the *MiddleScript* that performs actions continually during the scene, and the *EndingScript* that cleans up when the scene closes.

In the rest of this section I concentrate on describing how the basic perturbation-compensation process of the system is implemented (cf. Section 4.2). The two main objects involved are the *GraphicObject* and the *Vector*. The Vector generates the forcefields that move the GraphicObject in its phase-space. Author or user can perturb the vector, and thereby change the equilibrium of the forcefields.

The GraphicObject is implemented as a point in a phasespace of one or more dimensions (recall Fig. 2.2). Sections of this hyperspace are "decorated" with *Resources*, i.e. material that determines how the object is manifested (picture, sound, and movie) when located inside the section (recall Figs. 2.5, 2.6, and 2.8 in Section 2.1). A Resource is an object containing a picture, a mask, a sound or a movie. All combinations are possible: some resources contain all three items, whereas others may only consist of a sound.

I have found this resource-concept very useful since it forces the designer to think *multimedia* as a unity instead of thinking pictures, sounds and movies separately.

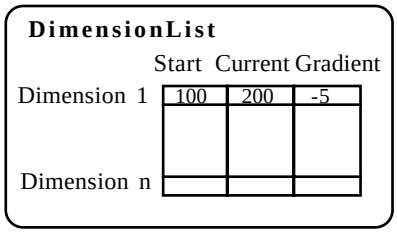

Fig. 6.2. *Dimensionlist* representing the location of the object in a hyperspace.

The location and trajectory of the GraphicObject is represented by means of a *DimensionList* that is a list of *Dimensions* consisting of 3 integers, *Start, Current,* and *Gradient*. The *Start* integer denotes the location of the object when it is generated, the *Current* denotes its current location, and the *Gradient* represents its trajectory in the sense that it indicates the next displacement of the object. When the Gradient (-5) is added to the Current (200) we get the next location of the object (195).

The main variables of the GraphicObject are shown in Code 6.1. The DimensionList is the variable *thespace*.

```
DimensionList thespace;
  // the space consisting of dimensions
  // in which the object resides
HyperSpace *hspace;
```

```
  // indexes shapes in one or more dimensions
PointIndex *rsrcdimension;
  // rsrcdimension->theindex[i] is the dimension
  // indexed for at the ith level
ResourceIndex *shapes;
  // index of available multimedia resources
StageRoom *thestage;
  // the StageRoom to which this object belongs
ActantSpec manifestation;
  // the Actants manifesting the object and the canvas
  // where they live

ActantSpec manifestation;
  // the manifestation of the graphicobject: the actant manifest-
   ing the object and the canvas where they live
```

Code 6.1

Only Vectors can move a GraphicObject in its space; the Vectors are controlled by different objects, on the one hand the user himself, on the other hand Events, Vectors or Constraints designed by the author. All objects influence the GraphicObject by means of gradients, and the trajectory of the object is determined by the sum of these gradients. This is one way of implementing multiple and conflicting forces in an interactive system.

The “decorated” phase-space sections are implemented as a tree with one or more levels referenced by the *hspace* variable in the object.

Fig. 6.3 shows the tree belonging to the View-object (a subclass of GraphicObject) in one of the modules. The View-object is the visible part of the background, and is used as the iris of the eye in Section 2.2 (recall Fig. 2.6). As we remember, the View must manifest different sounds, depending on its location. The top level of Fig. 6.3 describes the horizontal part of the section, and the bottom level the vertical part. Together they specify a rectangle of the background. In addition to coordinates, the bottom nodes of the tree indicate the resource that must be used in the section defined by the numbers of the path above them.

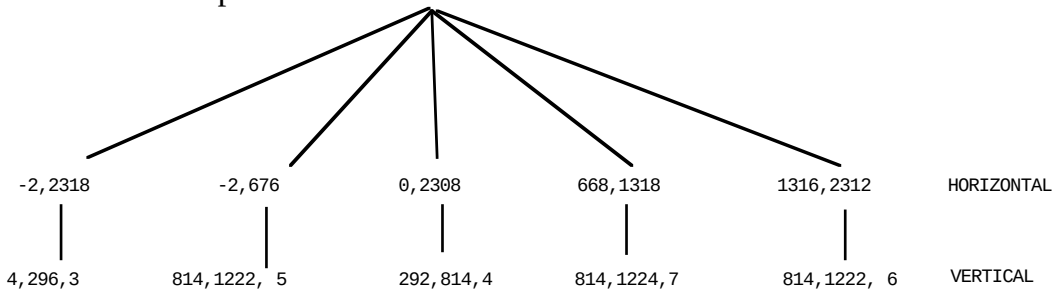

Fig. 6.3. The hyperspace of the GraphicObject.

Let us take an example. We want the View to be manifested by a resource containing the sound “lark-song” whenever it is located inside the rectangle “-2, 2318, -4, 296”, where the first pair of numbers denotes the left and right, the second pair the top and bottom of the rectangle.

The first thing we have to do is to build a hyperspace with a horizontal and a vertical dimension:

```
Stage(EYEOFWODAN)->aview->AddRsrcDimension(HORIZONTAL);
Stage(EYEOFWODAN)->aview->AddRsrcDimension(VERTICAL);
```

This is recorded in the *rsrcdimension* table of the object.

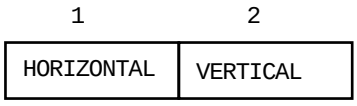

Fig. 6.4. The mapping between dimensions and levels of the tree *(rsrcdimension).* Level 1 indexes the horizontal, level 2 the vertical dimension.

saying that level 1 of the tree indexes the horizontal, level 2 the vertical dimension.

Then the sound resource “Lark” must be registered and recorded as a resource of the View.

```
ViewOf(EYEOFWODAN)->AddResource(3, ResNo(rLAERKE));
```

In this case it becomes resource number 3 of the View. Finally we enter the rectangle into the tree and indicate that this section should manifest the View by means of the resource “lark-song”:

```
ViewOf(EYEOFWODAN)->AddPath("-2,2318,-4,296", 3);
```

The leftmost branch in Fig. 6.3 shows the result of these actions.

This scheme is reasonably flexible — the number of dimensions is completely free — and can be made to work fast. In other cases the same machinery can be used to represent and manifest for example the mood of a person (Fig. 2.5 or Table 3.1) or the angle from which an object is seen (Fig. 2.8).

Let us now take a look at the object that move a GraphicObject, namely the *Vector*. Like all other objects, the vector is activated via its *Idle*-method. The vector implementing the pseudo-gravity potential in Section 4.1 creates a potential whose origin is located in relation to a source object *srca*, and lets a target object *targ* be influenced by the gradient of the potential. It is possible to indicate a particular point *(srcx, srcy)* inside the source object for the origin of the potential, and similarly a point inside the target object *(targx, targy)* as the onset-location of the force. The *height* and *width* of the vector

controls the shape of the forcefield: negative height yields an attractor, positive height a repeller. See Fig. 6.5 - 66.

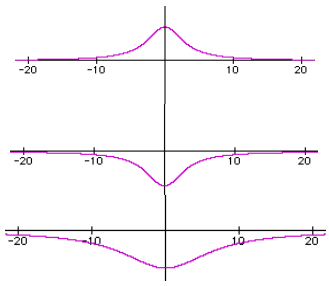

Fig. 6.5. Narrow repeller, narrow attractor, wide attractor.

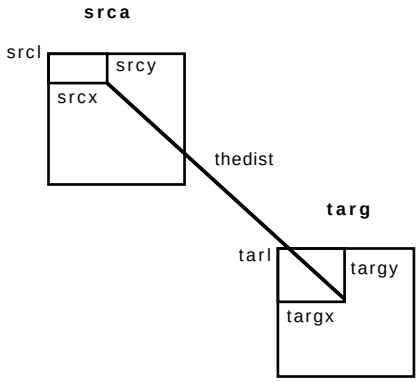

Fig. 6.6. Source and target of vector.

The gradient of the onset-location generated by the potential located at the origin is calculated, and the *x* and *y* displacements, *dx* and *dy*, are found. Finally, *dx* and *dy* are accumulated in the Gradient variable of the Dimension-list of the target, and the target marked as needing updating. All vectors work in this way but of course use different kinds of force-fields.

When the *Idle*-method of the GraphicObject is subsequently activated, it discovers that there is a non-zero gradient in one of its dimensions, and that consequently the object must move in its phase-space. It then searches the tree in Fig. 6.3 to determine whether it should take on a new shape in the new location, or the old one is still valid. If the former is the case, i.e. if a new resource has been found, it picks out the new parameters of manifestation (new horizontal and vertical location, new picture and mask, new sound and movie) and asks the objects that manifest it to realise these items (the *manifestation* variable in code 6.1).

The boundary between the “aesthetic” part (the “form”) and the part taking care of the manifestation is kept as clean as possible, and the two parts were in fact programmed by two different people. Apart from portability, the sharp boundary makes it easier to handle features with a similar aesthetic function that just happens to be manifested in different hardware

Time-limitations sometimes prevented us from taking advantage of this feature. A good example of this is the room. The room can be controlled in two ways: we can select silhouettes on the picture wall (Fig. 3.5 and 3.6), and we can control the intensity and colour of the spots in the ceiling. The simplest way of doing this is to represent the room as just another GraphicalObject, whose manifestation happened to be properties of spots and projected silhouettes instead of pictures on the screen. As indicated in Table 3.1, the room is a point in a Fullness-dimension, and can be moved in that dimension in the

same manner as all other GraphicObjects. The only difference is its manifestation. However, the room was implemented so late that we did not have time for revising the manifestation component, and we had to control the room with direct brute force. This again made it difficult to integrate the changes of the room with the rest of the system.

The main control loop is the *Idle*-method of the Theatre that does not do anything else than activate the *Idle*-methods of the system objects placed in various queues (e.g. Vectors in *forcequeue*, GraphicObjects in *actorqueue*, events[1] in the four *narrativequeues,* etc.). See Code 6.2.

```
void Theatre::Idle()
{
 succeeded = false;
 forcequeue->RunEm();
 // activates Idle-method of Vectors
 interactionqueue->RunEm();
 // activates Idle-method of EddaWindow. Handles user-interaction.
 if (authorenabled) {
  currentact->Idle();
  // activates Idle-method of Act. Reverts to default scene
  //  in the absence of user interaction
  narrativequeue->RunEm();
  // activates Idle-method of Events with priority 1
  if (! succeeded) {
   narrativequeue2->RunEm();
   // activates Idle-method of Events with priority 2
   if (! succeeded){
    narrativequeue3->RunEm();
    // activates Idle-method of Events with priority 3
    if (! succeeded)
     narrativequeue4->RunEm();
     // activates Idle-method of Events with priority 4
   }
  }
 }
 constraintqueue->RunEm();
 // activates Idle-method of Constraints
 actorqueue->RunEm();
 // activates Idle-method of GraphicObjects
}
```

Code 6.2

[1] The four queues for events is the way modality was implemented, events in $narrativequeue_i$ being more possible than those in $narrativequeue_{i+1}$. Impossible events are removed from the queues. Although this solution is efficient, it misses completely the generalisation inherent in using a modality dimension.

Summarising, we can say that the Vectors keep moving the GraphicObjects in their phase-spaces, and the GraphicObjects keep manifesting the new position in that space.

User interaction is handled in the *Window* that is part of the interaction-queue and contains one or more Canvasses that analyse the interaction (mousedown, mouseup, drag, click, etc.) and take suitable action. If the user has clicked inside the View, it sends a *Command*-message to a *StageRoom* object containing two vectors called *theauthorforce* and *theuserforce*. Both vectors will move the View but are disabled in the beginning; however, when the user has clicked at location *x,y, theuserforce* is activated and its origin set equal to x,y, i.e. the vector is moved to the place pointed to by the user.

Thus, the user is not allowed to move objects directly, but only to perturb the *uservector* (recall Section 4.1).

The author, who makes his contributions through his *Narratives,* lives under similar conditions. Narratives are recorded in the Act-object and contain one or more Events. Events contain three parts, a *trigger*, a *success act*, and a *failure act* (cf. Section 5). Apart from moving GraphicObjects in their hyperspace and thereby present or move pictures, sounds and movies, Triggers and Events can also influence the *authorforce* controlling the View. Many Triggers check whether the View is near the "scene of the crime" of the event; if this is true, the event does not immediately perform, but first activates the *authorforce* and sets its origin to the desired place. The *authorforce* then begins dragging the View towards the place, and, if the user allows it to succeed, the View reaches its destination, and the event can play. Thus, interaction is implemented as a battle of forcefields between user and author (Section 4.2).

Note that the system is based on pseudo-parallel processes because more than one narrative are typically active at time, so events from different narratives can be intertwined as in American police TV-series. Manifestation is also parallel in that more than one change can be manifested simultaneously.

Besides the origin of the forces, their height and width can be changed. A possible application of this feature is the following example from the development phase that was implemented by not actually used in the final system: the user is placed in a Viking village looking at the many beautiful things brought home by the Vikings. In this phase, there is a weak attractor keeping the user inside the boundaries of the village (in those time the village equalled safety). However, as user sees more and more beautiful objects, the attractor turns into a repeller, gently nudging the user out of the village to foreign countries. The idea is that the user partakes in the longing for news, loot, and honour that drove the Vikings themselves out on raids. Processes of

this kind are similar to Thom's catastrophes, where one set of equilibrium conditions changes into another one (Section 4.2).

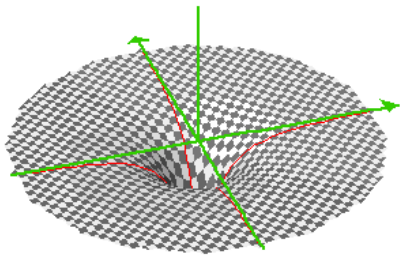

Fig. 6.7. At home in the village.

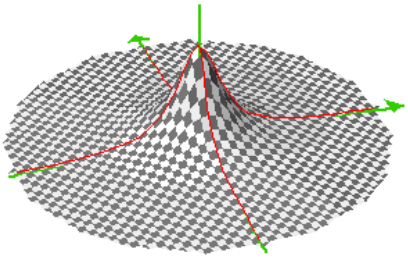

Fig. 6.8. The call of foreign places.

7. Conclusion

In this paper I have suggested the notion of dynamic phase-spaces as a possible way of designing and implementing interactive multimedia systems. The main advantage is that interaction, i.e. the balance of power between user and author, is built into the very foundations of the system.

The method of "decorating" the phase-space with resources that in principle contain all media, i.e. pictures, music, effectsounds, speech, and movies, discourages the designer from focusing on the individual medium and invites him to think of them as a unity.

In addition, it seems possible to describe more multimedia concepts by means of phase-spaces than originally expected, for example the room and the narratives.

However, the newness of the concept can be a disadvantage too, since its aesthetics is largely unknown. The project described in this paper took 2 years from the initial brainstorm until a finished system was put into operation. During this time, we had to realise that even if the concept offers many new possibilities, the user-contracts necessary for utilising them did not yet exist, and we therefore had to squeeze the system more into existing contracts than anticipated.

References

AARSETH, E. J.(1994). Nonlinearity and Literary Theory. In Landow 1994, 51 - 86.

AARSETH, E. (1995) *Cybertext*. Doctoral Dissertation, University of Bergen.

ANDERSEN, P. BØGH (1992). Vector spaces as the basic component of interactive systems. Towards a computer semiotics. *Hypermedia* 4(1), 53-76.

ANDERSEN, P. BØGH (1995a). The force dynamics of interactive systems. Towards a computer semiotics. *Semiotica* 103-1/2,5-45.

ANDERSEN, P. BØGH & B. HOLMQVIST (1990). Interactive Fiction. Artificial intelligence as a mode of sign production. *AI and Society **4**,* 291-314.

ANDERSEN, P. BØGH , J. W. JOHANNSEN & J. A. MIKKELSEN & M. SAMS (to appear). Interaktive tekster [Interactive texts].

BRANDT, P. AA. (1989). Agonistique et analyse dynamique catastrophiste du modal et de l'aspectuel. *Semiotica* 77-1/3.

BREMOND, C. (1966). La logique des possible narratifs. *Communications* 8: 60-76.

BREMOND, C. (1970). Morphology of the French Folktale. *Semiotica* 2: 247-276.

CARLSEN, J. (1994). *Odin & Harddisken [Wodan and the Harddisk]*. Copenhagen: Centrum.

LANDOW, G. P. (1994). *Hyper/Text/Theory*. Baltimore & London: The John Hopkins University Press.

LAUREL, B. (1991). *Computers as Theatre* . Redwood City, CA: Addison-Wesley.

MADSEN, K.H. (1994). A Guide to Metaphorical Design. *The Communications of the ACM*, vol 36 no 12 (57-62)

OGAWA, R., H. HARADA, & A. KAMEKO (1990). Scenario-based hypermedia: A model and a system. In: N. Streitz, A. Rizk, and J. André (eds.), *Hypertext: Concepts, systems and applications*, Cambridge: Cambridge University Press, 38-51.

THOM, R. (1983). *Mathematical models of morphogenesis.* Ellis Horwood: Chichester.

THOM, R. (1990). *Semio Physics. A sketch.* Redwood City, CA: Addison-Wesley Publ. Comp.

WILDGEN, W. (1982). *Catastrophe Theoretic Semantics*. Amsterdam: John Benjamins Publ. Comp.

WILDGEN, W. (1985). *Archentypen-semantik.* Tübingen: Gunter Narr Verlag.