A NEW METHODOLOGY FOR SPATIOTEMPORAL GAME DESIGN

Stéphane Natkin and Liliana Vega
Centre De Recherche en Informatique du CNAM
Conservatoire National des Arts et Métiers
292, rue St Martin - 75003 Paris, France
E-mails: {natkin, lvega}@cnam.fr

Stefan M. Grünvogel*
NOMADS Lab
Nonlinear Media: Art, Development and Science
Piusstr. 40, D-50823 Köln, Germany
E-mail: gruenvogel@nomadslab.org

KEYWORDS

Game design, spatiotemporal relations, Petri net, reachability tree, hypergraph, hyperedge replacement, connection

ABSTRACT

We propose a new formal approach for the design process of computer games, which involves the modeling of spatiotemporal relationships. Logical and temporal transactions are modeled using Petri Nets and topological relationships of the game universe by hypergraphs. For splicing these structures, we introduce *connections* which relate hyperedge replacement with the reachability tree of the Petri Net. By using these constructs flexible changes and the validation of certain properties of the missions can be accomplished.

INTRODUCTION

Game design is a difficult task that combines artistic and technical processes. Considering the interactive nature of computer games and, especially, their main goal (to entertain), the author must leave controlled freedom to the player (Bates 2001). The player has to be confronted with complex problems which are neither too easy nor too difficult to solve, making sure that game experience leads to a succession of goals within a reasonable time (Gal et al. 2002). In addition, the player must have a feeling of freedom in this interactive world, even when he is guided towards a solution in an unconscious way. To accomplish this, the game designer has to create *"one or more causally linked series of challenges in a simulated environment"* (Rollings 2003). This is not an easy job, as the following example shows.

Example: (room-key error) Two rooms *A* and *B* are only connected by a closed door, the avatar of the player is located in room *A* and its task is to get into room *B*. To achieve this he has to unlock the door with the help of a key. But the key is located in room *B*, thus the avatar will never be able to open the door.

Although this seems to be a trivial example, in real games different tasks can change the topology of the virtual space in a quite complex way, such that it is not easy to prevent the game designer from these or similar errors.

In this article, we concentrate on the modeling of spatiotemporal relationships of games. We start by giving a short overview of common design practices. The differentiation into game and level design is reflected by the two following sections. Special Petri Nets are used to define the missions of a game, and hypergraphs for characterizing the topology of the virtual space. To bring these two structures together we define after that *connections* and describe their basic applications.

GAME AND LEVEL DESIGN

In this article we consider the design process of a computer game from industry's classical point of view. This process is decomposed into two phases: Game Design and Level Design. For the description of both a common method is to use game design documents. These documents define the different elements of game design and illustrate the game concept: scenario, game and level missions, character description, etc. (Bates 2000, Rollings 2003). It becomes *the* reference document for all production team members.

Game creation starts in most of the cases by an original idea and the development of its scenario. Thus in a first conceptional stage the principal aspects of the game universe are defined (Gal et al. 2002): Epoch and style, context, goals to be reached, main type of objects involved, users game perception, etc. The definition of the game at this stage is known as the *Game Design*.

Level Design is the next specification stage. A game level consists of a virtual space, puzzles, main actions and a set of objects to interact with in order to complete a given goal. The difficulty of puzzles can be defined by the geometry and topology of space, logic, action sequences and objects localization. Each level has to be meaningful for the game, and goals have to be the central element unifying the level theme (Bates 2001). With this in mind, a game designer's task is to motivate the player by balancing objects behavior and game rules. Thus he has to take into account at each stage the space description, the positioning of objects in the virtual world, the logic of actions sequences relating these objects and the constraints caused by the topology of the virtual world.

* The author wants to express his sincere gratitude to Prof. Natkin and the CEDRIC-CNAM for the invitation to a research visit as *Professeur Invité* at the CEDRIC-CNAM during summer 2004.

PETRI NETS

We use Petri nets to model the ordering of action sequences in a game. This approach allows us to describe the logic structure of the missions in the game. The advantage of using Petri Nets (PN) is that they can be represented either by graphical or mathematical models, depending on model complexity and application context. (Peterson 1981). The following discussion is based on (Natkin and Vega 2003) approach.

PN game models are composed of transactions (cf. Figure 1). Transactions represent the atomic actions of the player, i.e. only one transaction can be executed at a time. Transactions can be in one of tree states: not started, executing or finished.

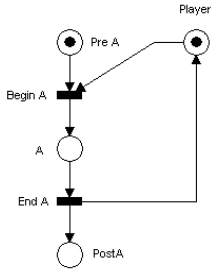

Figure 1. Basic Model of a Transaction.

A transaction net N = *(T,G)* is a set of transactions *T*, which are combined by three basic constructs. A relation between two transactions is denoted by *G*. Transaction nets describe possible choices the player has during the game. Given two transactions ***a*** and ***b***, they can be combined as shown in Figure 2. Each construct represents the possible sequence in which ***a*** and ***b*** can be executed. In Figure 2, the left construct show the case where ***a*** and ***b*** *are not related* (i.e. they can be executed in any order). In the center construct ***a*** *is before* ***b*****,** which means that ***b*** can only be executed if ***a*** has been executed**.** In the right construct if ***a*** is executed ***b*** cannot be executed.

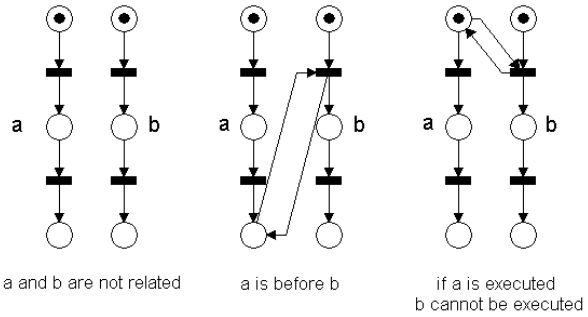

*Figure 2. Relations between two Transactions **a** and **b**.*

By combination of constructs, more complex semantics can be generated e.g. mutual exclusion between two transactions (i.e. if one transaction is executed, then the other one can not).

The reachability tree of a Petri Net generates a Petri Net language (cf. Diaz 2001). A sequence of firing transitions generates the corresponding strings. The reachability tree of a transaction net describes all possible sequences of atomic player-choices. In transaction nets, transactions are atomic, thus we can merge the *Begin* and *End* transition of a transaction into one letter, generating the language L*(N)*. Applying these concepts to the transaction net on Figure 2, the left construct in Figure 2 generates the strings ***{a, b, ab, ba}***, the middle construct ***{a, ab}*** and the right construct ***{a, b, ba}***.

Example: The former approach is applied to Silent Hill 2 (Konami 2002), a horror-adventure game that takes place in a mysterious and almost deserted town. The player controls an avatar (James) with unclear goals at the beginning. To accomplish his mission, he must explore the environment, fight against enemies and collect objects (keys, notes, information, etc.) to solve puzzles. The game starts at a parking bathroom outside town.

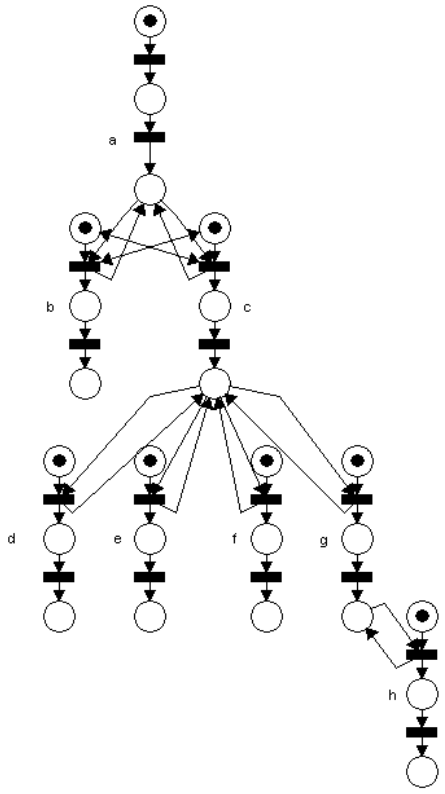

Figure 3. Silent Hill 2 First Level Transaction Net.

The main action sequence the player needs to execute to complete the first level is explained next (cf. Figure 3):

(a) *get the map from James car* and hit the road to Silent Hill. Once in the town, (c) *win the fight against a creature* in order to be able to continue. (b) *Loosing the fight* means that the game is over. After winning the fight, James can execute the next four actions in arbitrary order:
(d) *Recover an unclear inscription* from a weird monument.
(e) *Look at the map in the trailer.*
(f) *Find a second map in Neely's Bar.*
(g) *Find the apartment key on a dead body.*
(d), (e) and (f) are optional actions that can be repeated an infinite number of times with out changing the course of the game. (g) is mandatory and marks the end of the first level. After this, James is able to (h) *enter the apartment* to begin the next level.

SPATIAL RELATIONSHIPS

Besides describing the mission's logical structure (cf. the previous section), describing the game world topological properties and its evolution is also important.

There are different approaches to describe spatial relationships. One is to use visual languages (Haarslev, 1996) where description logic is used to combine topological and spatial relations and applications can be found in geographical information systems. A common way for describing topological relationships is to use the usual point-set topology with open and closed sets. In (Haarslev, 1996) an overview of different concepts is given, where often the interior and the boundary of sets are also taken into account. We do not follow this fine grained typology, but use instead a much simpler concept where we do not take into account the boundaries of sets.

In (Flury et al. 2003) an overview of location models in pervasive computing is given, where they establish the term *locus / loci* for location entities (e.g. regions of physical space) and *locants* for locatable entities, which can be passive or active objects (like a user). We adopt these terms and will denote a region of the virtual space as *locus*. In our approach, we do not model locants explicitly, but we concentrate instead only on the connections between different loci. The representation we introduce is a *locus graph,* where loci represent some subsets of space and their structural relationships are described by edges. These relationships are expressed here by (directed) hypergraphs.

Hypergraphs are generalizations of graphs, for a formal description and a thorough discussion of hypergraphs cf. (Drewes et al. 1997). For the presentation in the article, we take the somewhat simpler definition of (Gallo et. al. 1992). A *hypergraph H* is a pair *H=(V,E)*, where *V* is a finite set of *vertices* (or *nodes*) and $E = \{E_1, \mathrm{K}\ , E_m\}$ with $E_i \subseteq V$ for *i=1…m* is the set of *hyperedges*. Clearly, when $| E_i | = 2$ for all *i*, then the hypergraph is a standard graph. Additionally, by a finite nonempty set *C* of *labels* and a function $lab : E \to C$ the hyperedges can be labeled.

A directed hyperedge is an ordered pair *E=(X,Y)* of (possibly empty) disjoint subsets of vertices. *X* is the *tail* of *E* and *Y* its *head*. A *directed hypergraph* is a hypergraph with directed hyperedges, which we will denote hypergraph in the following for simplicity. Petri Nets can be modeled by hypergraphs and also (ordinary) directed graphs. Given a set of vertices *V* and labels *C* the class of all directed hypergraphs over *C* with these vertices is denoted by H(C).

To construct the topological relationships, the virtual space of a game is divided into loci, which are represented by nodes *V* in a hypergraph. Each locus is a pathconnected subset of the virtual space, which means that the avatar can move to every point in the locus at any time of the game if he is within the locus. Typical examples for loci are the rooms of a house, or special zones in an outdoor area. The set of loci does not change during the game, it is fixed. Therefore the partitioning of the virtual world into loci defines all the possible places of interest which have to be taken into account for the game. Note, that loci do not have to be maximal with respect of the pathconnection property, which means that it is e.g. possible to divide a room into two loci (the left and the right half) while the avatar is able to move freely within the room.

The ability for an avatar to move from one locus to another at a certain state of the game is modeled by an hyperedge E_i. Because the hypergraph we use is directed, it is possible to model one-way connections, where it is only possible to move an object from one locus (the tail of the hyperedge) to another (the head), but not the same way back. Note also, that the hypergraph is not unique because there may be several ways of defining the corresponding hyperedges. We use hyperedges because they have more possibilities for presenting spatial relationships than edges from ordinary graphs, which can only connect two nodes.

While playing the game, the hypergraphs may change, i.e. certain hyperedges can be replaced by others or even vanish. Formally this can be described by hyperedge replacement cf. (Drewes et.al. 1997). As mentioned before, we assume that the decomposition of the virtual space into loci is fixed and does not change during the game.

Example: In Figure 4 a map of the first part of Silent Hill 2 is given (non-real scale).

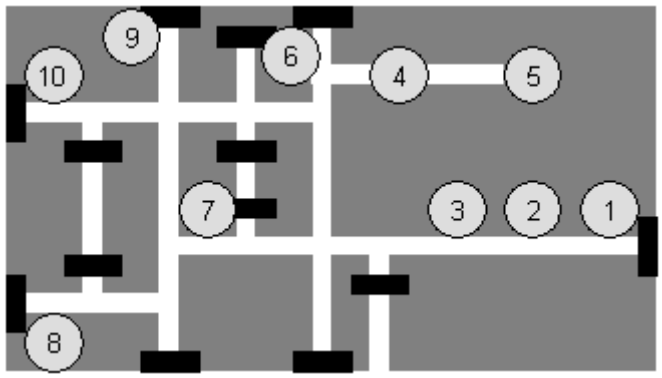

Figure 4. Partial Map of Silent Hill 2. White: Street, Grey: Building/Area, Black: Roadblock, 1 Parking Place, 2 Fountain, 3 Church, 4 Backyard, 5 Tunnel, 6 Swamp Monument, 7 Neely's Bar, 8 Trailer, 9 Apartment Gate Key, 10 Apartment.

The white stripes represent the streets, the grey areas buildings or open areas. The circled numbers represent special subsets of the virtual world where the player experiences special actions or has to fulfill specific tasks. These are taken as nodes in the hypergraph of Figure 5, where the corresponding spatial relationships between these loci at the beginning of the game are represented. Hyperedges are represented as arrows with the label of the hyperedge within a square. Note, that the hyperedge *A* and *A'* are not connected with any nodes (especially not with node *1*), which reflects that at the beginning of the game, the

protagonist can not leave the parking place. For graphical representation, we also gather several hyperedges into the edge *D*, which means that you can move freely between any of the nodes *3* and *4*.

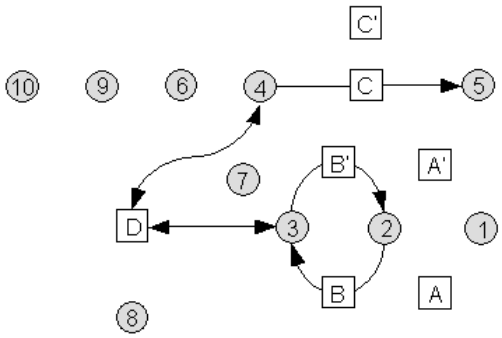

Figure 5. Spatial Relationships at the Start.

CONNECTIONS

The change of the topology of the virtual world can be modeled by hyperedge replacement. For a formal definition of general hyperedge replacement, we refer to (Drewes et.al. 1997). For the sake of simplicity we only introduce in this article the replacement of one hyperedge at a time. But all of the following constructions are also possible for general hyperedge replacements on hypergraphs.

Let the set of vertices *V* be fixed, *C* a set of labels and H(C) denote the corresponding set of hypergraphs over *C*, let $A \subseteq V$ be a hyperedge to be replaced by another hyperedge $B \subseteq V$. Then the *replacement [A/B]* is a function *[A/B]* from H to H, where a hypergraph *H* is mapped to the hypergraph *H[A/B]* by removing *A* from *H* and adding the hyperedge *B* to *H*. If the hyperedge *A* is labeled, then the new hyperedge *B* inherits this label.

As an example, the hyperedge *e = ({},{})* with *lab(e) = A* in Figure 5 does not connect any node. It is replaced by the hyperedge *e'= ({1}, {2})* with *lab(e')=A* in Figure 6, which actually connects now the nodes *1* and *2*.

In the following we introduce *connections* relating hypergraphs with transaction nets.

Definition: Let *V* be a set of vertices, *C* as set of labels, H(C) be the corresponding set of hypergraphs and N *=(T,G)* a transaction net with a corresponding language L*(N)*. Let $\hat{T} = T \cup \{*, ?\}$ be the augmented set of transactions. Then a *connection* is a pair *([A/B], p)* where *[A/B]* is a replacement and $p \in \hat{T}^*$. Here $\hat{T}^*$ is the set of all strings of $\hat{T}$ including the empty string. We call *[A/B]* the *replacement* and *p* the *pattern* of the connection.

Now the semantics of a connection *([A/B], p)* is defined as follows. Given the overall mission by a transaction net N and the initial spatial relationship of the mission by a hypergraph *H*. By walking through the transaction net, a string *s* of increasing length is created. If the string s contains the string *p* then the corresponding replacement *[A/B]* is applied on the hypergraph. The wildcard symbols "*" and "*?*" have the usual semantics known from common pattern matching programs, where "*" means "arbitrary symbols" and "*?*" "at most one symbol".

Example: We consider again the Silent Hill example. After having examined the car, the avatar is able to leave the parking lot. This can be modeled in the following way. Let *W=({},{})* and *X=({1},{2})* with *lab(W)=lab(X) = A* and *Y=({},{})* and *Z=({2},{1})* with *lab(Y)=lab(Z) = A'* be hyperedges. Then the connectors are defined as

([W/X], a) and *([Y/Z], a),*

where *a* is the transaction from Figure 3.

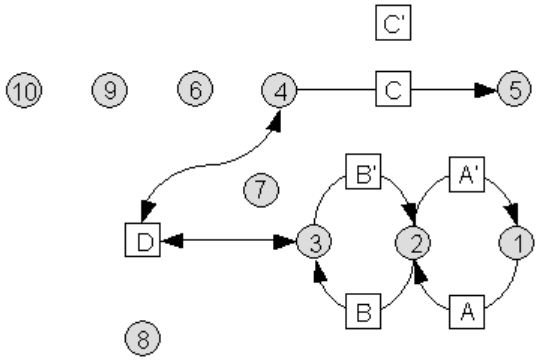

Figure 6. After investigating car in (1), the player can leave the parking place.

In the same way, that only after killing the monster in the tunnel the avatar is able to leave the tunnel is modeled by

([K/L], c) and *([M/N], g),*

with *K=({},{})*, *L=({5},{4})* and *lab(K) = lab(L) = C'* and *M=({3,4},{3,4}), N=({3,4,6,7,8,9}, {3,4,6,7,8,9})* and *lab(M) = lab(N) = D.* (cf. Figure 7).

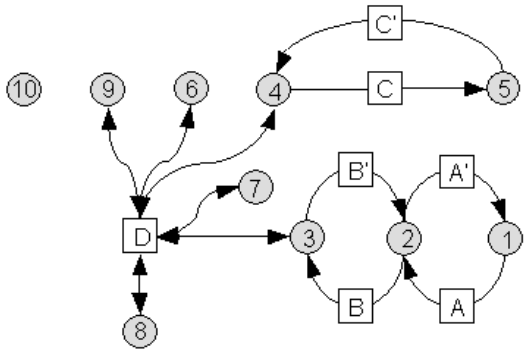

Figure 7. After the fight with the monster is won.

Finally the accessibility of the apartments after receiving the key (cf. Figure 8) is expressed by

([P/Q], g)

with *P=({3,4,6,7,8,9},{3,4,6,8,9}),* and *Q=({3,4,6,7,8,9,10}, {3,4,6,8,9,10})* and *lab(M) = lab(N) = D.* (Note that for

presentation reasons we collapsed the nodes of several hypergraphs into *D*).

Given a transaction net, an initial hypergraph and some connectors, it is also possible to add some additional conditions on the connectors, e.g. that the pattern of a connector appears in every word of the transaction net, or in at least one. This can be easily verified by standard algorithms on Petri Nets.

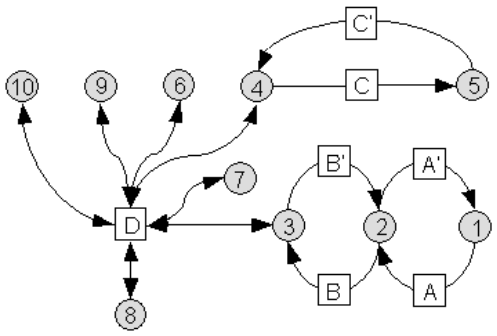

Figure 8. After getting the key to the apartment.

These instruments can be used for new methodologies in the game design process. One method would be to start with the map of a mission and define the replacements corresponding to certain transactions. After these are defined by connectors, the overall logic can be developed by connecting these transactions or adding some additional ones to build the overall transaction net. The opposite way is also possible: start with a transaction net, construct a map and define the connectors. In general, game design is a non-linear process which will follow neither of the ways in a strict manner. For this case it is also possible to switch forth and back between changing the transaction net and the connectors or even the map during the design.

The formalism can also be used to validate certain spatiotemporal relationships. To test if certain nodes in the hypergraph are connected after specific transactions where activated, one has to create the corresponding hypergraphs for all the words where the transactions appear, and test with standard algorithms (cf. Gallo, 1993) if the nodes are connected.

To test if certain nodes are connected at all, one has to check, which kind of patterns of the connectors actually can be found in the language of the transaction net, create the corresponding hypergraphs and check for connectivity.

Also the room-key problem of the introductory example can be solved, by checking if two nodes are connected before one applies a connector. More general, one can also think of defining a sequence of sets of nodes and a sequence of transactions where the interconnections of the nodes are conditions for the firing of the transactions.

CONCLUSIONS AND FURTHER RESEARCH

We introduced a new methodology for combining spatial and temporal relationships in games. For the modeling of the spatial relationships we used hypergraphs. But because Petri Nets also can be modeled as hypergraphs, it would be interesting supplying hypergraphs also with semantics of Petri Nets. In this case it has to be investigated, what actually *then* could be modeled by the tokens in the spatial hypergraphs and how this Petri Net can be connected to the transaction net in a meaningful way.

Although in principle it is possible to express multiplayer missions, the current framework has to be extended to support the special needs of multiplayer games (e.g. how to support cooperation between players). This is also related to the question, how other objects in the game (e.g. equipment, weapons or power-ups) can be represented in the framework.

For an evaluation it is thought of implementing a test environment. However, the task of providing game and level designers with an appropriate and intuitive user interface has to be taken into account.

REFERENCES

Bates, B. 2001. *Game Design: The Art & Business of Creating Games*, Prima Publishing.

Diaz, M. 2001. *Les Réseaux de Petri*, Hermes, Paris

Drewes, F.; A. Habel; and H. Kreowski. 1997. "Hyperedge Replacement Graph Grammars", In *Handbook of Graph Grammars and Computing by Graph Transformation. Vol. 1: Foundations*, G. Rozenberg editor, World Scientific, 95-162.

Flury, T.; G. Privat and N. Chraiet. 2003. "A Model and Software Architecture for Location-management in Smart Devices / Ambient Communications Environments", In *Communication with Smart Objects: developing technology for usable pervasive computing sysems*. Kintzig,C; Poulain, G.; Privat, G.; Favennec, P. (eds.), Kogan Page Science, 71-90

Gal, V.; C. Le Prado; S. Natkin and L. Vega. 2002. "Writing for Video Games", In *Proceedings of the VRIC 2002*, June 2002, Laval, France, 245-252

Gallo, G.; G. Longo; S. Pallottino; and S. Nguyen. 1993. "Directed Hypergraphs and Applications", *Discrete Applied Mathematics*, Vol. 42 , Issue 2-3, 177-201

Haarslev, V. 1996. "A Fully Formalized Theory for Describing Visual Notations", In *Proceedings of the AVI'96 Post-Conference Workshop on Theory of Visual Languages*, May 30, 1996, Gubbio, Italy.

Konami. 2002, "Silent Hill 2", computer game, developed and published by Konami.

Peterson, J.L. 1981. "Petri Net Theory and the Modeling of Systems", Prentice Hall, New Jersey.

Natkin, S. and L. Vega. 2003. "A Petri Net Model for the Analysis of The Ordering of Actions in Computer Games", In *Proceedings of the GAME ON*, London, November 2003.

Rollings, A. and E. Adams. 2003. *Andrew Rollings and Ernest Adams on Game Design*. New Riders, Indianapolis.