Towards a Graph-Based Model of Computer Games

Iwona Grabska-Gradzińska[1], Bartosz Porębski[2], Wojciech Palacz[3], Ewa Grabska[4]
Faculty of Physics, Astronomy and Applied Computer Science, Cracow, Poland
Email: [1]iwona.grabska@uj.edu.pl, [2]bartosz.porebski@uj.edu.pl,
[3]wojciech.palacz@uj.edu.pl, [4]ewa.grabska@uj.edu.pl

***Abstract*—This paper proposes a new holistic approach to a formal model of computer games. The story and structure of a computer game is represented by a hierarchical layered graph, meanwhile the way that the game is played – by graph transformations. This approach enables comparative description of different games, analysis of dependencies between a game structure and players' strategies, automatic gameplay generation, and switching from single- to multi-player mode.**

***Index Terms*—game design, gameplay analysis, game plot modeling, hierarchical graphs, graph transformations, narrative structures**

I. Introduction

This paper proposes a new formal description of computer games. The main goal is to encompass the most important game aspects in one holistic model. This approach is promising in both practical and theoretical problems of computer game design.

Methodology of game research was developed with traditional games in mind. Rules of ancient games are barely known – they are reconstructed on base of preserved tokens, boards, pictures and literary descriptions. Rules of more contemporary games were created in times when all rules had to be remembered by one man and all dependencies between game elements calculated in mind. Nowadays computers completely change the way the games are played. NPCs (Non Player Characters) with artificial intelligence, complicated rules, sophisticated modifiers – all that can be computed by machine, which increases speed of opponent's reaction, rapidity of environmental changes and, in general, number of stimuli. Therefore the old XIX/XX-century based discourse of game research is not sufficient for modern computer games. It does not give the holistic view and describes only single, separated game aspects.

Computer games are mainly described by data structures of a specific game engine. Usually, the following aspects are considered separately: game as application, plot and narrative structures of the story game is based on, status of players and their roles in the formation of the game-play (scope of their decisions and flexibility of the plot) [1]. As a consequence – it is hard to represent structure of the game as one model including all aspects of playability. Computer scientists are interested in algorithms, programmers in implementation details, sociologists in players' emotions, motivations and engagement in virtual reality, literary critics in plot structure. For a long time games were in spectrum of interest of sociologists (as a social activity) and narratologists (if there is a story inside gameplay) because no computer was need to play [2]. Times changed and now the huge branch of game industry is based on programmers' creations.

The problem is that different game aspects influence each other significantly. Lack of holistic view causes that sometimes even beautiful visual-sophisticated games have pointless stories and boring, unbearably artificial characters, or very interesting story in beautiful environment gives players only frustration because game reality rules are not intuitive. The present challenge is to describe and measure players behavior and dependencies between player strategies and gameplay, player impressions and, at last, effectiveness of the game understood as "wow factor".

A. Rules

The heart of gameplay is a set of rules which enable players to make decisions, reach the goal, and get the prize [3]. As long as games were human-refereed, the rules could be unclear, sometimes inconsistent and often being modified *ad hoc* during the activity. Computer games changed this situation. Computer as a game master needs precision and consequence in preparing gameplay. That is the cause that many computer games are very schematic and no spontaneous reactions are allowed.

The problem is that on one hand there is immeasurable variety of plots, stories and narrations, and on the other hand a list of scripts and schemas of behavior has to be short.

B. Narrative elements

Not only game designers, but also book publishers dreamed about automatic plot generation, but the great variety of events which happen in everyday life made it seem like a futile dream. On the other hand, the frequent *déjà vu* effect implies that story tellers used only a small fragment of the spectrum of narrative possibilities. Literary theory explains that paradoxically all range of human behaviors can be described by small amount of narrative units. The early twentieth century work of Russian structuralist [4], Vladimir Propp gives a proof that construction of any folktale of Russian culture tradition is based on eight types of character: the villain – struggles against the hero; the donor – character who makes the lack known and sends the hero off; the (magical) helper – helps the hero in the quest; the princess or prize – the hero deserves her throughout the story but is unable to marry her because of an unfair evil, usually because of the villain. The hero's journey is often ended when he marries the princess, thereby beating the villain; her father – gives the task to the hero, identifies the false hero, marries the hero, often sought for during the narrative.; the dispatcher – prepares the hero or gives the hero some magical object; the hero or victim/seeker hero – reacts to the donor, weds the

DOI: 03.CSS.2012.3.27

princess; false hero – takes credit for the hero's actions or tries to marry the princess. These characters take sequence of 31 actions: absentation; interdiction; violation of interdiction; reconnaissance; delivery; trickery; complicity; villainy or lack; mediation; beginning counter-action; departure; first function of the donor; hero's reaction; receipt of a magical agent; guidance; struggle; branding; victory; liquidation; return; pursuit; rescue; unrecognized arrival; unfounded claims; difficult task; solution; recognition; exposure; transfiguration; punishment; wedding [4]. On the higher level of abstractive thinking every story can be built from these narrative units, like a puzzle. This regularity provides a great possibility to create computer aided game design structure. Manipulation of finite set of elements produces complete spectrum of storytelling.

The problem is that generating the plot for the novel means finding one of typical sequences of character actions which obeys rules and choose the environment which authenticate or, at least, not disturb the story. Generating the gameplay means preparing the environment to support every story which fulfills the rules [5]. Rules must include all aspects of history and give us answers for all questions.

C. Why graph-based model?

Computer game is a kind of virtual reality – simulation of elements of real world. It is system of high complexity, but, on the other hand, built with components quite well categorized and somehow organized. There are four main types of components:

- **characters** (human-like creatures with a set of human-like properties),
- **locations** (areas where characters' actions take place),
- **items** of environment,
- **narrative elements**: plot, rules, character development paths, behavior patterns.

They are connected geographically, chronologically, and, most importantly, influence each other. Especially plot elements stimulate and restrict interferences between other elements.

Game is a fascinating combination of flexibility and regularity. Right proportions of these both components give a mind-blowing gameplay. Missing any of them means that game is too boring or too complicated to play intuitively.

The most convenient way to show complexity of real or virtual world is a graph model. The universal system of nodes and edges is a tool to describe majority of dependencies, hierarchy, structure.

This mathematical approach is novel, because so far the natural language, not formal structures, has been used for description and analysis of game narrations.

Moreover, a graph structure allows one to notice unexpected correlations. The well-defined graph network shows similarities between phenomena we never associated before and explain seemingly non-explainable behavior of examined system, for instance correlations between players' behavior and structure of the gameplay can be analyzed.

II. Example

Based on the categorization by Arthur Asa Berger [1] game structures can be divided into following types: simulations, real-time strategy games, first person shooters, action arcade games, adventure games, role playing games, sport games.

Concept and rules of sport games are directly taken from reality. The challenge is how to increase illusion of world around and impression of tactility.

In all of these types there are additional differences between single-player and multiplayer structures, but the plot can be considered in both cases as the same structure, just multiplied in the latter case.

The origin of game industry is an innate need to compete. Term "to win" is shared by many languages as one of primaries in human relations. Playing games is often related to practicing skills, empathy and abstract thinking. The oldest games, let us call them "situational", placed emphasis on this first usability. "Who will first scoop up the stone into that hole, One, two three, start" – and there is the first game: with game-players, environment, object, rules of their manipulation and winning conditions. In the course of time rules became more sophisticated, environment more abstractive and hierarchy of game-players more complex. Shortly speaking it is the way from slinging stones at each other to practicing chess playing.

Human need to compete is a primal mechanism of improving skills. Not only physical, as in race, but also social and emotional achievements. This type of "psychological" games needs not only rules but also a plot – to find analogies between game and reality. This game function is not very far from story-telling. Listening to the myths, legends, folktales and stories helps to define social roles, identities, casual connections and social behavior. Games based on story-telling became extremely popular. Everybody can be a knight, a king, a princess or a dragon and learn what decisions, emotions and responsibilities are connected with this particular social role. It is much more effective social training than just listening. Plot-based games are nowadays the broadest part of game industry.

The following part of this paper focuses on this type plot-based adventure games.

Let us take into consideration a typical adventure game. Long time ago there was a kingdom led by a righteous king. He had a beautiful daughter. One time a huge disaster threatened the kingdom – a gargantuan dragon started to burn houses, kill people and animals.

Brave knights tried to fight but they failed. Terror grown. The terrified king announced that his daughter would marry the man who defeats the dragon.

The main game task is to save the world from the dragon. Getting married to the princess and becoming a king is an award for the winner.

DOI: 03.CSS.2012.3.27

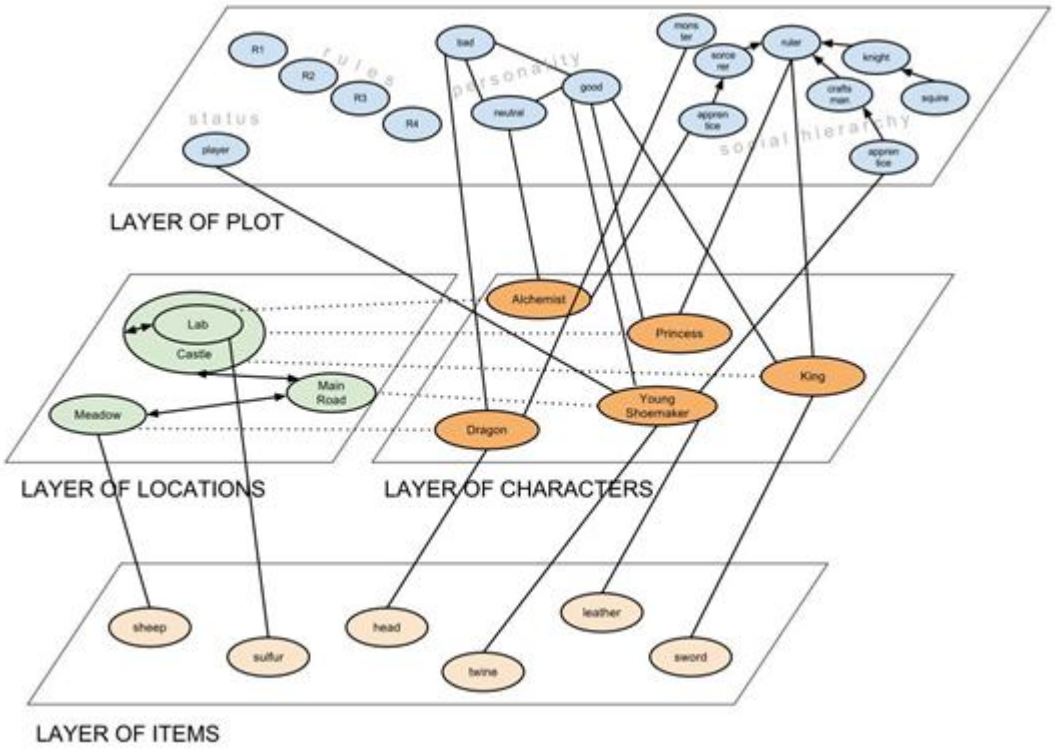

Figure 1. Structure of the game

A. Structure of the world

Let us imagine this kingdom: the castle, few cities being repeatedly burnt by the dragon, the main road and forests around. These are locations which characters occupy. To such a kingdom fit characters of few types: rulers (in this role King and Princess), knights, craftsmen (including our player: Young Shoemaker), peasants and one monster represented by Dragon. There are items: possessed by characters or placed in locations.

The structure of the game is shown in Fig. 1. The world is described by rules (denoted as R1, …, R4) and dependencies, which define interpersonal relations, societal hierarchy and behaviors of human and animals. For example: monsters like eating sheep and young virgins, people die when attacked by a monster, monsters die after eating sulfur.

Poisoning with sulfur-filled sheep carcass is a standard procedure for getting rid of dragons in Polish folktales.

Each character is described by elements of a plot (nodes on the highest layer), place he occupies (a node in the loca tions layer) and items he possesses (nodes in the lowest layer). As focus of our interest is character we can notice that sub-graphs describing character have specific shape, like a sheaf.

Structure of character information is shown in Fig. 2. On the left: a character (orange node) determined by three elements of plot (blue ones) is in a specified location (green one) and owning two items (beige ones). That sub-graph stands for Young Shoemaker having twine and leather, walking along road and obeying three rules: he is a craftsman, he is good and he is a player character. In the middle there is another location with two items in it. It is a meadow and two sheep. On the right: another character determined by some plot elements and having one item. That sub-graph stands for Dragon.

As Fig. 3 shows, characters' activity can be modeled by changing the graph: Dragon changed location to the meadow (tempted by sheep, as we can presume) and the Young Shoemaker did the same. When two characters of opposite nature (good and bad ones) find themselves in the same location, an edge representing a fight is established between them.

The most useful tool to transform graph is to define productions changing one sub-graph into another. For any further manipulation a formal definition of the graph system is needed.

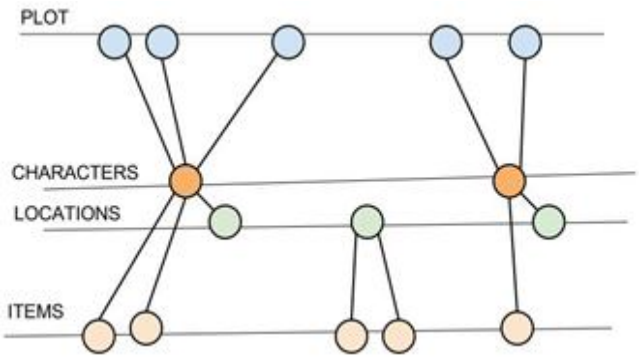

Figure 2. "Sheaf" sub-graph describing character's situation

DOI: 03.CSS.2012.3.27

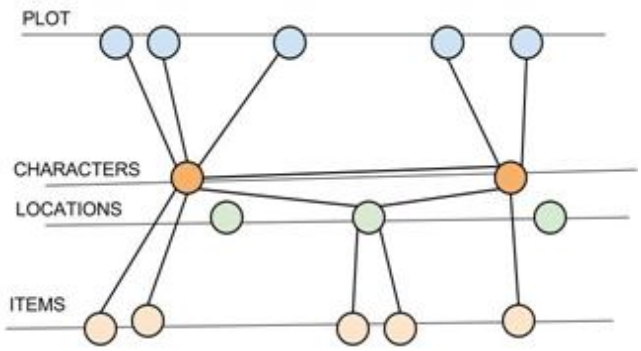

Figure 3. Sub-graph from Fig. 2 after the change

III. Formalism

A mathematical definition of graphs is needed for precise understanding of their capabilities and as a basis for future implementation of the game engine. Many different definitions can be found in the literature; this problem needs graphs which are labeled and attributed, can represent both symmetrical and asymmetrical relations, allow for nesting of nodes in other nodes, and for separation between different node types (plot elements, characters, locations, and items).

The definition used in this paper is based on [6,7]. The hierarchical graphs allow for straightforward modeling of compound objects, i.e. a military squad consisting of several soldiers, or a sheep with a sulfur filling. Nodes are connected by directed edges (asymmetrical), but a pair of edges which connect the same two nodes going in opposite directions can easily be used to represent a symmetrical relation. Labels and attributes provide information about real-life objects which are being modeled. Graph layers allow for separation of nodes into four subsets.

Let L be an alphabet of labels for nodes and edges. Let A be a nonempty, finite set of attributes. For every $a \in A$, let D_a be a fixed, nonempty set of its admissible values, called the domain of a. If f is a function, then let f^+ be its transitive closure.

A **hierarchical graph** is a system
$G = (V, E, s, t, ch, lab, atr)$,
where:

- V and E are finite disjoint sets of nodes and edges,
- $s : E \to V$ and $t : E \to V$ are edge source and edge target functions,
- $ch : V \cup E \to 2^{V \cup E}$ is a child nesting function such that $\forall x \in V \cup E : x \notin ch^+(x)$,
- $lab : V \cup E \to L$ is a labeling function,
- $atr : V \cup E \to 2^A$ is an attributing function.

This definition provides attributes, but does not determine what their values are. It can be said that nodes and edges in these graphs are generic.

Let us come back to the game. There can be a node which represents a sword, with attributes for damage dealt and its price in gold pieces, but it is an abstract sword – it is not known if it is a +50 sword of dragon slaying which costs ten thousand, or a worthless −1 rusty sword. A graph with concrete values assigned to its attributes is known as an instance.

A **graph instance** is defined as
$I = (V, E, s, t, ch, lab, atr, val)$,
where:

- $(V, E, s, t, ch, lab, atr)$ is a graph,
- $val : (V \cup E) \times A \to D$, with $D = U_{a \in A} D_a$, is a partial function such that

for all $x \in V \cup E$ and $a \in A$
if $a \in atr(x)$ then $val(x, a) \in D_a$.

Let us assume that L (the set of labels) consists of three disjoint subsets Σ_V, Σ_D, and Σ_U, which are used to label nodes, directed edges, and undirected edges. Further-more, let $Y = \{Y_1, Y_2, \ldots Y_n\}$ be a partition of Σ_V.

Graph G or instance I is layered if and only if
for all $x \in V$
$lab(x) \in Y_i$ implies that $lab(V)'' ch^+(x)) \subset Y_i$
(i.e., a node and all other nodes nested in it have labels from the same layer).

In the specific case presented in this paper
$Y = \{PLOT, CHAR, LOC, ITEM\}$.

Summarizing, a gameplay graph is, formally speaking, a layered graph instance. Fig. 4 shows an example of such a graph model.

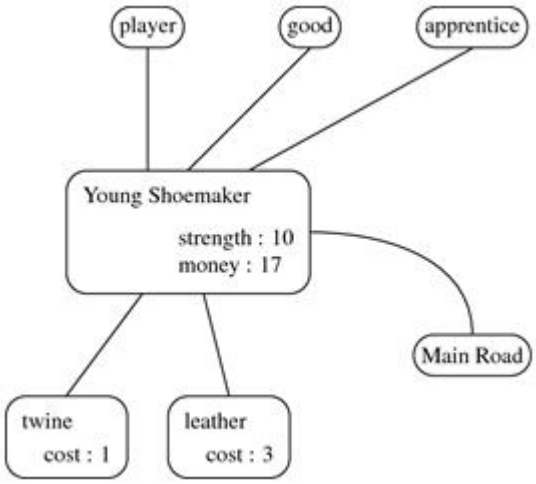

Fig. 4. A graph instance corresponding to the left part of Fig. 2

Actions undertaken by game characters change gameplay graphs. We would like to use graph productions to represent these actions, because this is the way usually used to formally define the process of graph rewriting.

Productions used in this paper are based on single-pushout productions [8]. They were extended to handle hierarchical graphs, attributes, and to be able to match not only nodes with a specific label, but also nodes with any label belonging to a specific layer. This ability is extensively used to specify actions which operate on any character, or at any location.

A production consists of the left- and the right-hand side, the application condition, and the family of value assigning functions. Both sides are graphs (not graph instances), and there is a partial morphism from the left side to the right side. Nodes and edges mapped through this morphism are called "the context", and are preserved when the production is applied.

DOI: 03.CSS.2012.3. 27

The process of applying a production to a graph instance begins with searching the instance for a fragment which matches the left-hand side. After the match is found, the application condition is evaluated. If no match could be found, or if the condition was not fulfilled, then the whole process fails – the production cannot be applied.

Otherwise, the process continues. First, it removes from the instance fragment those nodes and edges which are matched to the non-context part of the left-hand side. Next, it adds to the instance copies of non-context nodes and edges from the right-hand side. Finally, it evaluates the value assigning functions, which specify attribute values for just added nodes and edges (they can also modify attribute values in the context nodes and edges).

IV. Player Activity

The player knows that in general he wants to get from the world where he is closer to Dragon than to the Princess to the situation where Dragon is killed and Princess is his wife.

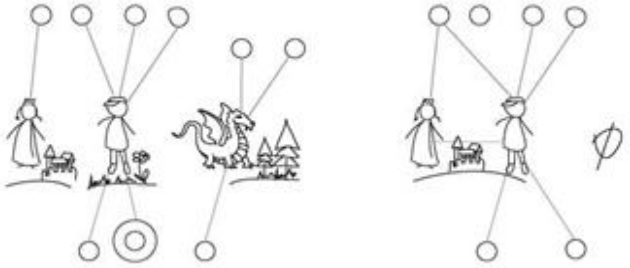

Figure 5. Starting and winning gameplay states

Let us consider the main, most typical activities defined by the simple productions.

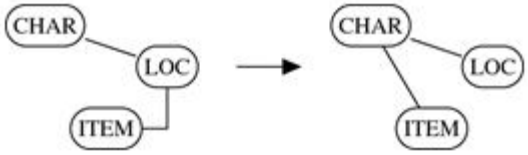

Figure 6. Picking up an item

The first example, displayed in Fig. 6, represents the action of picking up an item. Please note that instead of specific node labels the production uses layer names; this means that the left-hand side can be matched to any character in a gameplay graph, which is currently in some unspecified location, where in turn some unspecified item is lying around.

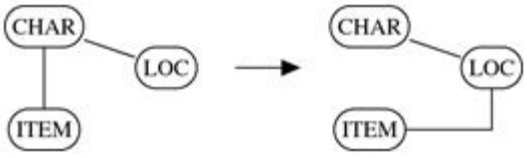

Figure 7. Dropping an item

Fig. 7 displays an opposite production, which describes the action of dropping an item.

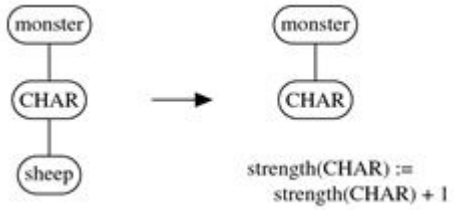

Figure 8. Eating a sheep

Fig. 8 illustrates operations on node attributes. Let us assume that all characters have a numerical attribute "strength". A monster which has eaten a sheep should increase its strength (of course only if it has not died earlier because of sulfur hidden in the sheep – see next example).

The left-hand side can be matched to any character which is a monster (i.e., it is connected by an edge to the "monster" node in the plot layer), and which has (as in "has in its stomach") a sheep item. The right-hand side does not contain a corresponding "sheep" node, thus the matched sheep and its edge are deleted from the gameplay graph. Then the instruction specified on the right-hand side is executed, and changes the value assigned to the "strength" attribute of the matched character.

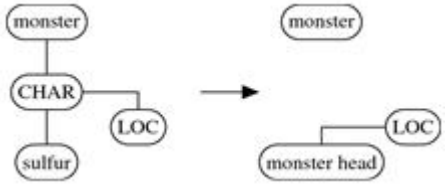

Figure 9. Death of a monster

Fig. 9 is also concerned with monsters. This time, a monster has eaten something which contained sulfur, and now has a piece of sulfur in its stomach. It dies of poisoning, leaving in the location of its demise a part of its corpse – the head.

V. Applications And Advantages

Using formal, graph-based model of gameplay structure should provide many advantages. Using graph rules to represent dynamic changes of the game state is the main one, as various dynamic changes are essential for games. Additional advantages include also some practical applications.

A. Aditional mechanism

Graphs are widely used in software engineering. There are many libraries and tools for graph processing. There are also many well-understood additional formalisms which extend their power or help with solving particular problems.

For example: control diagrams can be used for representing sequences of actions of Non-Player Characters (NPCs). Players' and NPCs' actions can be represented in the same way (as productions – mutual influences between the player and NPCs are significant), but with schemes of proceedings defined for non-playable elements of virtual reality.

The same mechanism can be used to prioritize selected actions. It helps to avoid an inappropriate order of actions

DOI: 03.CSS.2012.3. 27

expressed as graph productions.

B. Formal ways of proving properties

Representing gameplay models and their modifications in mathematical terms of graph grammars should provide an opportunity to formally reason about their properties. Some of them are of obvious practical importance.

For example, if a given graph rewriting system can generate an infinite language, then this is a game which can reach an infinite number of states, and thus can crash (because the computer running the game has finite memory). If the language is finite, but it is possible to construct infinite production sequences, then the game will not crash, but may never end. The game has no dead ends (does not become unwinnable in the middle of a play) if for every state where the player character is alive a sequence of productions which transforms the graph model into a state where the player wins can be found.

C. Automatic generation of different gameplays

Automation provides an opportunity to test new game concepts before resources are spent on creating new game and before trials with human testers are conducted. It can be checked if there is no useless loops, dead ends and other storyline mistakes.

D. Designing for single- and multiplayer game structures at the same time

Modeling structure of gameplay can be made easier by using graph-based models, because all character activities (for PCs and NPCs both) are modeled in the same way. Differences appear only on the level of control diagrams and not on the level of productions.

E. Comparative description of different games and transition from gameplay structure to metastructure of players' behaviors

By using formal models game researchers can make comparative description of game structures and search for distinctive differences between various categories of games. The proposed graph model gives them also capabilities to systematic description of both behaviors and decisions of players. In this approach winning a game means to find a sequence of productions which leads the player from the beginning to the victorious end. An example of this network is shown in Fig. 10. There are many ways of winning a game, if a plot is a real nonlinear structure.

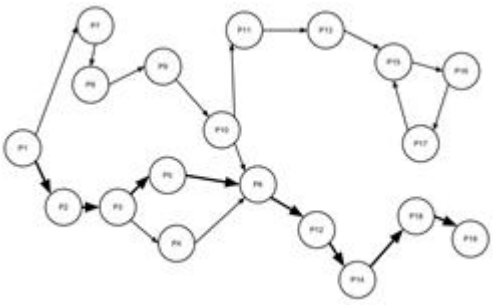

Figure 10. Traces of players' activities – network of production sequences

A linear game has one winning sequence with possibly some small loops and deviations. A truly nonlinear game has a very sophisticated structure of winning paths, but multitude of possibilities can be overwhelming for an average player – there are many loops and long paths between key productions developing actions. Graph and network tools can describe and measure all that features.

VI. Conclusions

The real limitation of using narrative structures is designer imagination and inclination to linear thinking and duplicating narration structures. It is hard to compose a coherent structure with many chances for taking part in the story. There are many possibilities of implementation nonlinearity and quasi-nonlinearity. There are problems with describing differences.

The proposed graph-based model is an attempt to help to overcome these limitations. It allows both game designers to organize and visualize dependencies in game structures and researchers to make description and analysis more precise and comparison of game structures easier.

Up to now promising simulation results based on similar graph-models have been obtained for CAD-problems [9] and the hp-adaptive Three Dimensional Finite Element Method [10]. The system engine for game structures is now being implemented. Simulation results for computer games will be the next step of the proposed approach.

References

[1] A. A. Berger, *Video games: a popular culture phenomenon*, New Brunswick, Transaction Publishers, 2002.

[2] M. Eskelinen, "The gaming situation", Game studies, vol. 1, no. 1, 2001.

[3] J. Huizinga, *Homo Ludens*, Boston, The Beacon Press, 1955.

[4] V. Propp, *Morphology of the folktale*, 2nd edition, University of Texas Press, Austin, Texas, 1958.

[5] E. Adams, *Fundamentals of Game Design*, 2nd edition, Peachpit, 2009.

[6] E. Grabska, W. Palacz, B. Strug, and G. Ślusarczyk, "A Graph-Based Generation of Virtual Grids", LNCS, vol. 7203, pp. 451-460, 2012.

[7] W. Palacz, "Algebraic hierarchical graph transformation", JCSS, vol. 68, issue 3, pp. 497-520, 2004.

[8] G. Rozenberg, *Handbook of Graph Grammars and Computing by Graph Transformation*, vol. 1-3, World Scientific, 1997-99.

[9] W. Palacz, E. Grabska, Sz. Gajek, "Conceptual Designing Supported by Automated Checking of Design Requirements and Constraints", D. D. Frey et al. (eds.), Improving Complex Systems Today, Advanced Concurrent Engineering, pp. 257-265, Springer-Verlag, 2011.

[10] A. Paszyńska, E. Grabska, M. Paszyński: "A Graph Grammar Model of the hp Adaptive Three Dimensional Finite Element Method", Part I. Fundam. Inform. 114(2), pp. 149-182, 2012.

DOI: 03.CSS.2012.3. 27