Branched Structure Component for a Video Game Scenario Prototype Generator

Gulnara F. Sahibgareeva, Vlada V. Kugurakova

Kazan (Volga Region) Federal University, Kremlyovskaya str., 18, Kazan, 420008, Russia

Abstract

The task of automating the routine work of computer game writers, narrative designers, set forth in earlier works, has been continued in the presented work. The issues of visualization of branching narrative structures of computer games are considered, the analysis of various approaches to visualization of the plot and other important components of a video game, such as, for example, automatic balancing of quantitative parameters, is carried out.

The paper presents the chosen technological stack and gives specific solutions for storage in the form of a structured scenario, allowing the generation of continuing story branches and testing the narrative prototyping stage using the automatically generated text novel.

Keywords

Interactive storytelling, computer games, game script, visualization, branched structures, graphs, narrative prototyping, script prototype, GPT-2, ruGPT3, python, Unity

1. Introduction

The rapid growth of the video game market [1] also causes a natural increase in the number of intelligent systems for their development. Since the process of computer game development is a long and expensive process, game development consists of many separate phases: 2D or 3D visualization, UI/UX design, gameplay design, artificial intelligence programming, character, level and environment design, scenario creation, narrative design, sound and music, and game logic programming itself. Therefore, it is important for the industry to create new effective tools to automate routine processes.

Analysis of such tools [2] shows that they contribute to increasing the level of variability in the plot of games:

"The use of artificial intelligence in the implementation of interactive narrative systems increases the expressive power of the system by partially assuming creative responsibility for the narrative experience of the user. This, in turn, can provide greater responsiveness and narrative diversity without reducing player autonomy."

In this work, we will focus our attention on visualizing the branching structure of the story, checking the constructed plot graphs for consistency, the ability to store a simplified (compared to natural text) structured scenario in JSON format, and the automatic generation of continuation story branches. All these new features should be integrated into the overall solution for working on the interactive narrative of computer games [3, 4].

2. The graphic representation of the narrative

To automate the process of approval of computer game scenarios, a high-quality visualization of its branched structure with the possibility not only of its automatic construction, but also of automatic logic checking is necessary.

A number of applications, such as Twine [5], Articy:Draft [6], Fungus [7], Storybricks Engine [8] implement, to some extent, game content management functionality, including the display of structures.

SSI-2021: Scientific Services & Internet, September 20–23, 2021, Moscow (online)
EMAIL: gulnara.sahibgareeva42@gmail.com (G.F. Sahibgareeva); vlada.kugurakova@gmail.com (V.V. Kugurakova)
ORCID: 0000-0003-4673-3253 (G.F. Sahibgareeva); 0000-0002-1552-4910 (V.V. Kugurakova)

For example, the Storybricks Engine, a mechanism with elements of artificial intelligence, formalizes the possibility of creating a narrative story that forms the basis of a future computer game, with extremely complex, branching story arcs.

As an example of the complexity of interactive structures we can give a vivid example (see Fig. 1) of a fragment of a scenario from the working project of Quantic Dream on the game Detroit: Become Human [9]. A completely unreadable representation nevertheless makes a lot of sense, but it is clearly necessary to change the display towards greater illustrativeness.

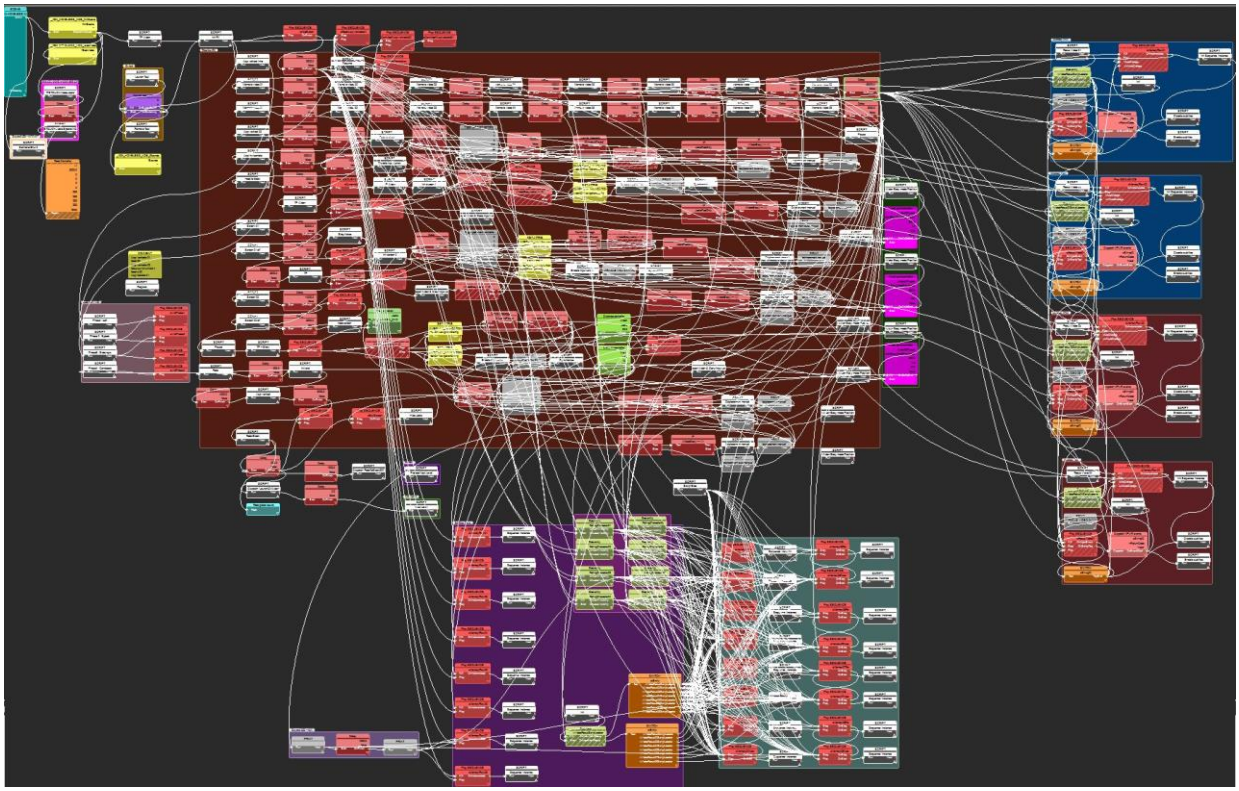

Figure 1: A part of the script for Detroit: Become Human [10]

By the way, the visualization of branching structures is well represented in scientific literature, often not at all related to the topic of game development. For example, StoryFlow [11] is used to specify the concretization of chronological events (see Fig. 2) in books or movie series – and these structures are represented as threads of yarn. In such yarns one can well reflect the variability of events taking place or the interaction of characters in specific time intervals.

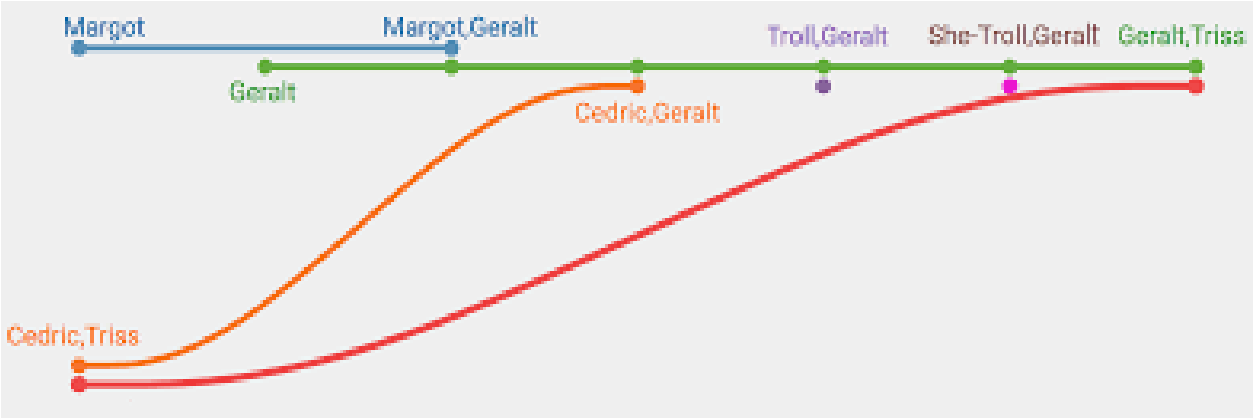

Figure 2: An example of some yarn structure

Another interesting way to visualize structures is flow diagrams (Fig. 3), in which the width of the arrows is proportional to the flow rate, the so-called Sankey diagrams [12]. This representation can help to

reflect the dynamics of specific data, as an example of use in video game development we can suggest tracking the change in the characteristics of the subjects as the story progresses.

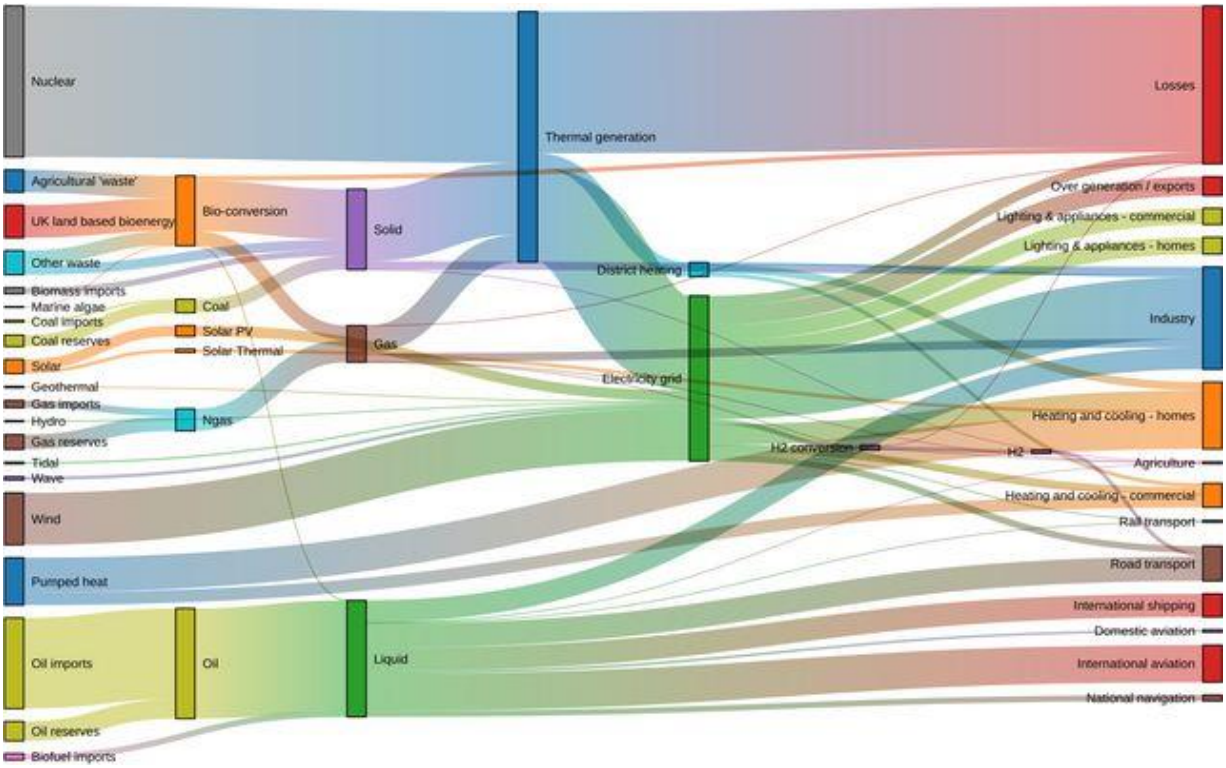

Figure 3: An example of a Sankey diagram

Another interesting solution for works of fiction that should be integrated into narrative design tools is the visualization of tonalities and entities drawn from the text. We implemented some such approaches using the python library Spacy, conducting experiments on texts from J.R.R. Tolkien's six-volume The Lord of the Rings, in which we extracted subject → relation → object triplets, named text entities (characters, locations, artifacts, etc.) and filtered triplets based on the found entities. The networkx library was used to build the graph, and the matplotlib library was used to draw it. The list of vertices-entities and edges-relationships was formed from the set of triplets obtained at the last stage of processing (see Fig. 4).

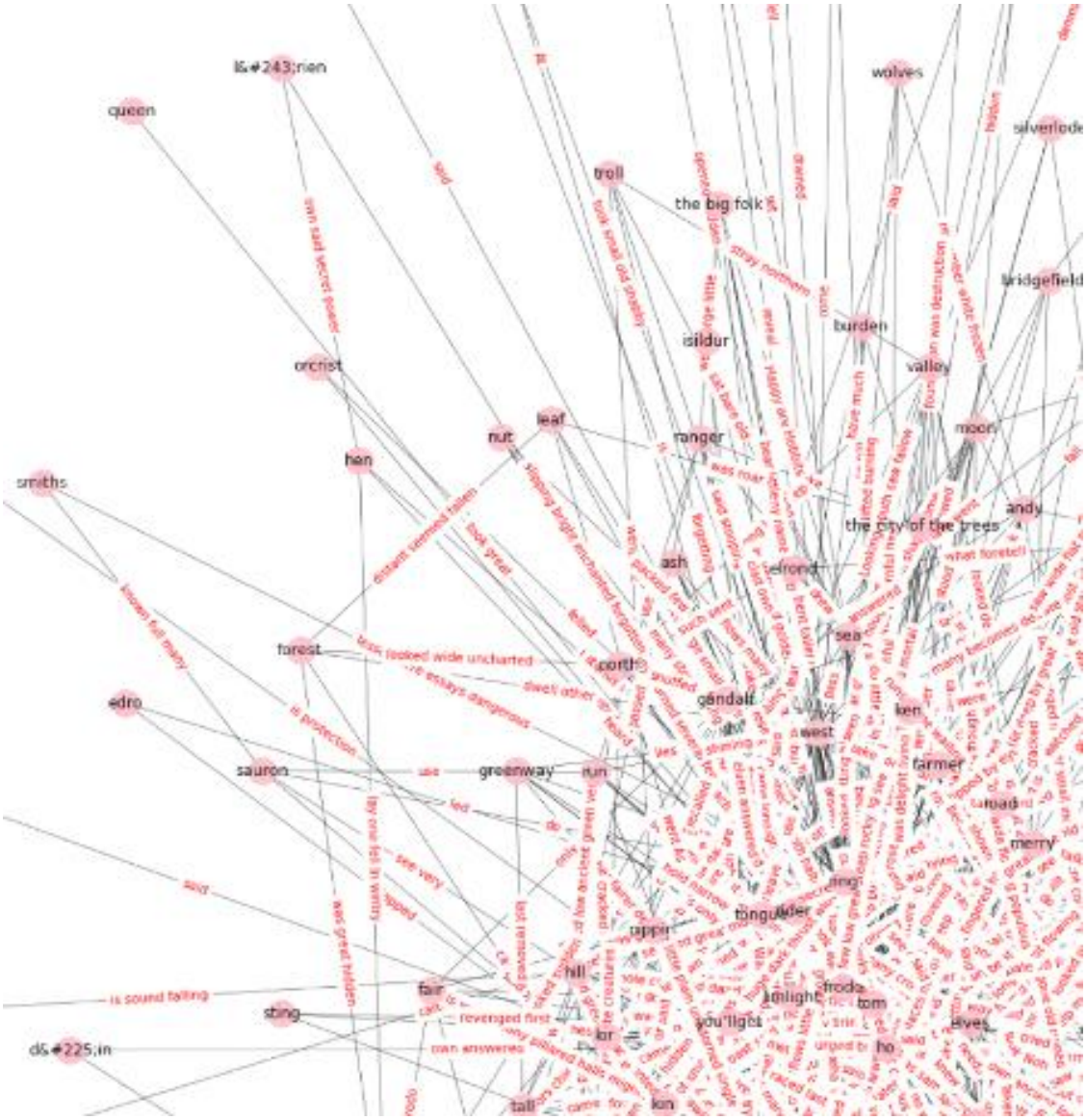

Figure 4: A fragment of the entity graph of *The Lord of the Rings*

Using the TextBlob library, the tone of the text for each sentence was detected. The obtained values of tonality and subjectivity were used as visualization parameters. Tone (values from -1 to 1) was interpreted as the deviation angle of the graph line, the value of subjectivity as the length of the graph line. Based on these parameters, the coordinates of graph points were calculated using the following formulas:

$$d = d + T\ (\pi / 2);$$
$$x = x_0 + \cos(d)\ p\ S$$
$$y = y_0 + \sin(d)\ p\ S,$$

where d is the direction (in radians, 0 by default), x_0 and y_0 are the coordinates of the previous point, p is the line length (constant), S is the subjectivity (values from 0 to 1), T is the tone (values from -1 to 1).

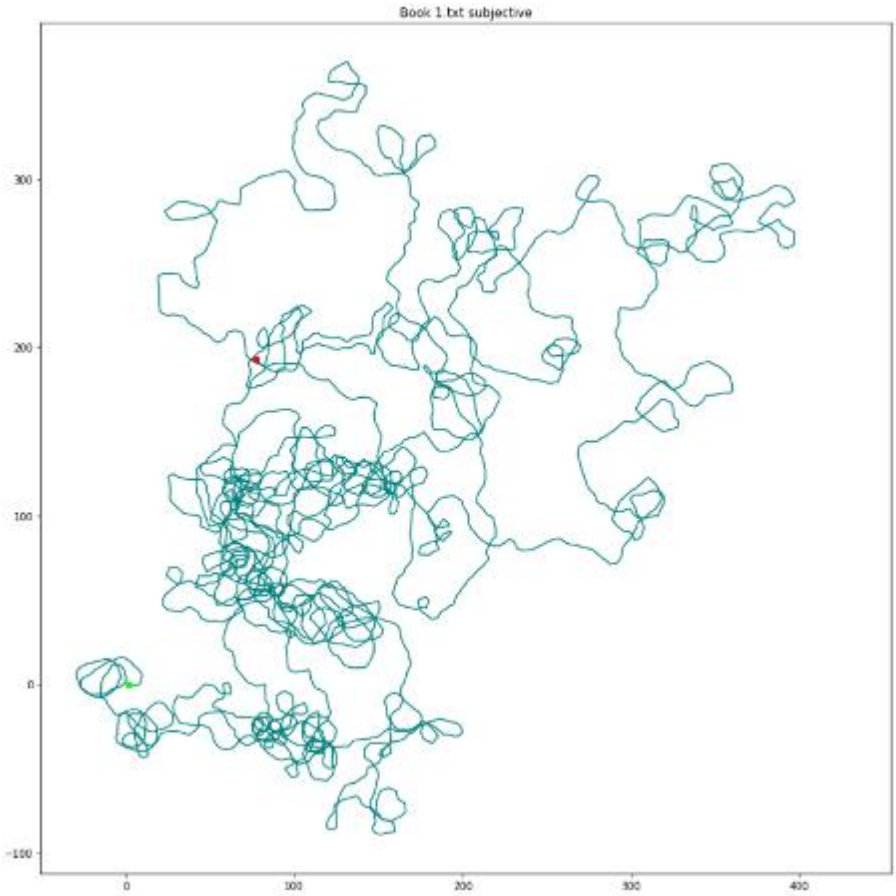

Figure 5: An example of visualization of the tone of the text of the first boof of *The Lord of the Rings*

In addition, it should be noted that there is a rather extensive classification of various structures [13] of video game scenarios, however, without limiting the generality we can say that these structures are common to any interactive experience (see, e.g., Fig. 6). It is logical that templates of such structures should be available to narrative designers when designing the narrative of a video game.

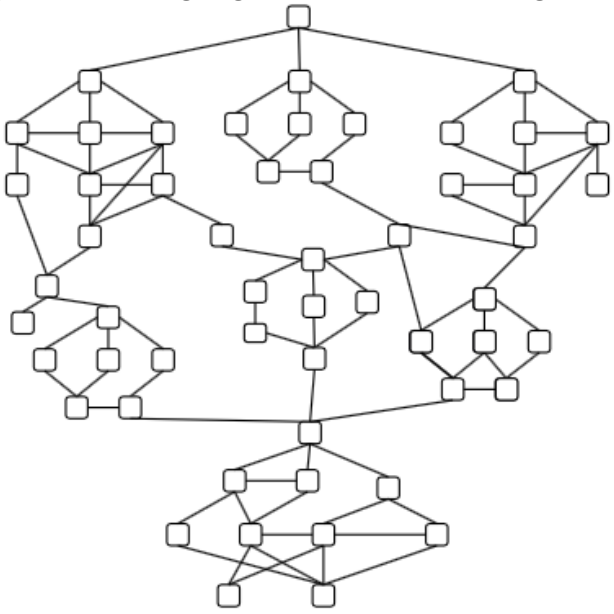

Figure 6: One of the branched out structures (Quest)

Without abandoning the future implementation of these and other forms of visualization, which may be appropriate to display data for private tasks, we have chosen as a representation a directed graph, the implementation of the plot by means of which, however, does not yet solve the problems arising for more complex structures, such as in Fig. 1.

2.1. Scenario Prototype Generation

Previous works described the general concept of a tool for generating a scenario prototype [3, 4, 14, 15]. Also the functionality of generation of camera positions and objects relative to each other by a text query was presented in a separate work [16].

Thus, the generalized work pipeline of the generator (see Fig. 7) looks as follows:

1. The script text in natural language is analyzed by algorithms, which extract from them information about in-game entities: names and characteristics of the characters, their lines, description of the location, the main events.
2. The information on the branched structure is visualized in a convenient form, statistics are given.
3. 3D scene is generated based on the received information, 3D models and animations are automatically selected.
4. Program interactivity is generated in the form of an opportunity to choose the transition to this or that event.

The assembly of the project is completed by the formation of an installation file, which is a scenario prototype, in other words, an interactive project that players and all interested parties can walk through or test.

Continuing with the development, we outline the necessary functionality to work out the new features:

1. formalizing the approach of storing a particular branching structure reflecting the video game's story;
2. allowing automatic continuation of the story;
3. automatic balancing of quantitative parameters;
4. generating the prototype as a narrative novella.

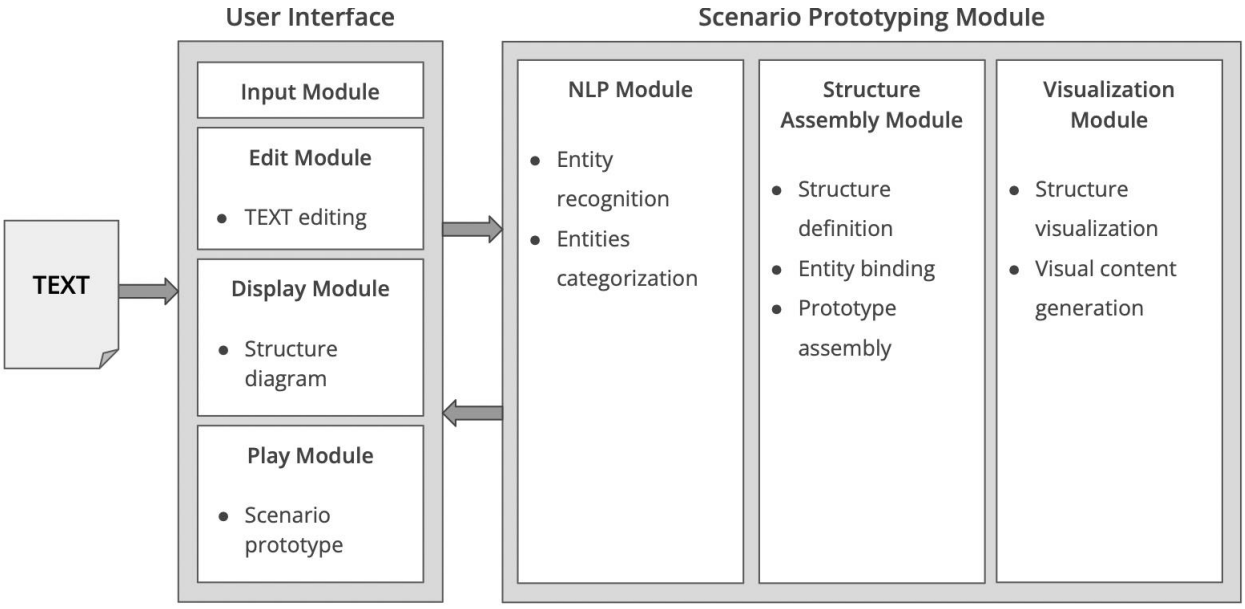

Figure 7: The architecture of the tool for generating a scenario prototype

2.2. The storyline branching problem

Another module allows generating additional vertices, i.e. the continuation of events, based on existing text passages and actions available to the player.

The idea was borrowed from the Storybricks Engine tool [8], which at one time was actively developing its own technological approach in the formation of a dynamic narrative. They proposed a narrative engine using artificial intelligence, which gives developers the ability to create and control narratives with very complex branching story arcs. First announced at the 2014 Game AI Conference, held in Vienna, the engine had the makings of fairly practical solutions to some long-standing problems of interactive story design: in particular replay ability – because most narrative games (such as BioShock Infinite [20]) follow a single story path that offers almost no replay ability.

Traditional narrative in video games works as follows: the developers try to create a story that unfolds based on the actions performed by the players. In this model, players in the game change the world by defining trigger conditions that reveal the next step in the story.

Using procedural content generation to automatically create branches where they are needed from character motivation bricks and the impact of that motivation on the game world, the StoryBricks AI tool generated a new storyline on the fly based on how player choices affect the motivations of others. We were impressed with this idea, and the development of neural networks made it easy to implement a similar approach.

To generate the continuation of the plot, the result of the GPT-2 (Generative Pre-trained Transformer) [21, 22] is used, which is a powerful language model that can be adapted to a wide range of NLP tasks using a very small set of task-specific data. Until recently, there were no such models for the Russian language. The AI Journey competition (https://ai-journey.ru, accessed September 2, 2021) released the ruGPT3 model, capable of generating coherent and meaningful texts in Russian as well.

The main feature of GPT-2 and ruGPT3 in particular is that the neural network does not need to be retrained for a specific task to show the results the user wants. The neural network adapts to the style and content of the text, which allows it to generate realistic passages that continue the original phrases. Immediately after training, the neural network is ready to generate a text with all the logical inserts: repeated mentioning of characters' names, quotes, references, excerpts of the same style throughout the text, a connected narrative.

It is worth noting that the algorithm generates a unique text every time, even with the same query (see Table 1). The example shows variants of continuation without semantic meaningfulness.

Table 1

Different variants of the generated text continuation

Artistic excerpt	Generated continuation
The camera turns slowly to the right, showing an elderly gentleman standing in a cave. Quick footsteps are heard as a hooded figure in a white robe runs up behind the man and forces him to kneel, then plunges a hidden blade into the back of his head, killing him.	The camera pans back, but from a different angle – the shadow of a man kneeling next to the corpse now appears.
	The killer's hand flies up, and the camera pans upward, showing a tall, gray-bearded priest lifting up one of the skulls on a long hilt.
	The camera goes to the left, showing another underground tunnel, lit by torches.
	The corpse falls to the floor.

In the future, more efficient and functional generation based on all character relationships should be worked out, including timing parameters that can be extracted from a full script rather than an excerpt.

2.3. The task of searching for inconsistencies

Separate attention should be paid to the function of searching for inconsistencies in the constructed scenario graph. These can be inconsistencies of character properties and transitions between vertices.

Each character is its name, as well as the values of the properties that make up the description of that character. Properties can be integer, logical or text values.

It is possible to automatically move to a vertex when an action is performed, or to move independently depending on the player's choice.

The vertex can check property values for conditions. If the condition is not fulfilled, the vertex is unavailable for the player. Example: a character has a text value of archer class. If the character has a different class, then a different pool of options is available to him.

In addition, in each vertex it is possible to apply functions to change property values depending on the events that occur. Example: an archer has an integer property “health”. If the character gets in trouble, this value decreases, which is done by appropriate calculations in the vertex.

2.4. The problem of balancing game values

Determining whether a game is balanced is not an easy task and is complicated by the often inaccessible mathematical relationships in game mechanics. However, it is possible to rely on statistics and assumptions. Players and developers of multiplayer games analyze the effect of individual game objects on winning or losing a game. Sirlin [23] views game balancing as the iterative task of bringing the game to a state where the options presented to the player are not only plentiful but also viable. One way of looking at balance is to treat it as a function of many variables, each of which represents a different aspect of the game that needs to be optimized. But as the complexity of the game as a system increases, difficulties arise in defining such a function or checking the current state of the game for optimality. However, formalization of these moments is necessary not only for automatic balancing, but also for manual balancing.

At the moment, there are different notions of "balance" [24]. This may be a consequence of the fact that different genres of games, or even different representatives of the same genre, naturally need a particular version of the concept of game balance. The most common definition proposed by Schreiber [25] is that game balance is basically about figuring out which numbers to use in a game.

Players usually have their own idea of balance. They find their own balance by examining different strategies and elements of the game to find the strongest ones. In this way, experienced players often have an advantage over newcomers. This state can of course be called a state of balance, but in this case there are often elements of the game that remain unclaimed. This leads to a potential narrowing of strategic options.

Developers can only adjust the parameters of their game entities, with no ability to directly influence player behavior. The only method to influence players is to optimize game parameters, as this forces players to adapt to the new environment and rediscover the new state of equilibrium [26].

So, optimizing balance in a game is a complex process, so game developers often approach this issue in stages, in small steps trying to bring the whole system to their definition of balance. In larger games, there is a more diverse choice of strategies and actions that the player can choose from. In such cases, a state of balance is almost unattainable.

When balancing a game manually, this process can be time-consuming, as each iteration of the adjustment must carefully analyze changes in balance. This has led to interest in automating this process. The following is a review of works on this topic.

The widespread use of machine learning algorithms has raised the question of what their potential is in solving the problems of game developers. Volz et al. [27] have applied it to creating game environments from a pool of existing mechanics and game sets using different artificial intelligence agents based on different notions of fitness to describe what makes a game balanced and fun. Different approaches have been used in these studies, and both have been successful.

The work [28] on describing an integrated game balancing process gives a better understanding of the problems of automating the balancing process, and illustrates ways to use such algorithms to solve game development problems. Among the conclusions of this paper is that artificial intelligence often does not play as a human would play. This leads to the fact that the data derived from the automated system cannot guarantee a balanced system for human players. This leads to the fact that developers need to check the balance in the game with their own players after receiving the results of the automated system.

In [29], genetic algorithms were applied to find balanced character skills for role-playing games. This work was done on a small scale, with large simplifications of the custom game model and a basic intelligence that made decisions based on the rules. Thus, this work does not consider the application of this methodology in the context of real games and in the presence of the influence of other players.

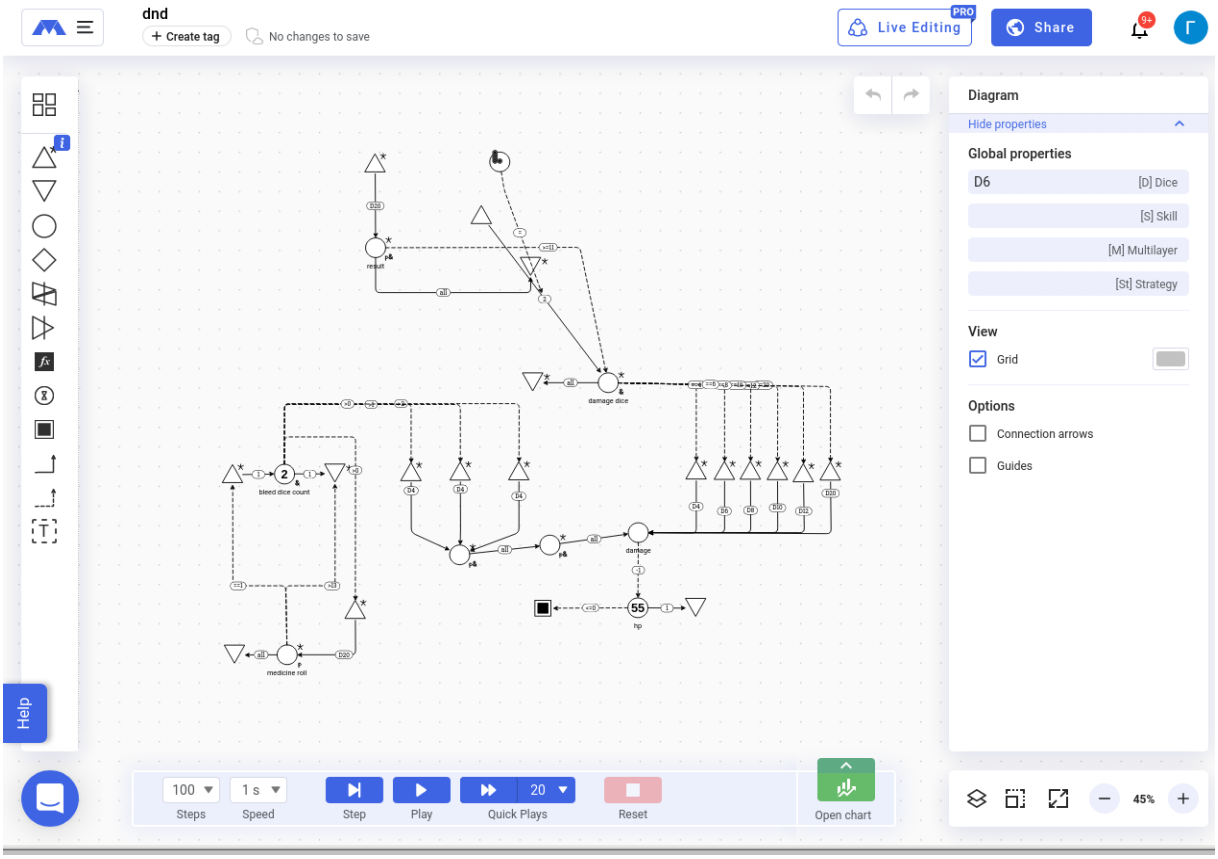

Figure 8: An example Machination diagram

In [30, 31] they investigated the applicability of DPBM[2] in the design of automatic game balancing systems. As a result of their work, they defined a conditional equilibrium condition, where there was some relation between classes, which defines for each class a list of classes that are easier or harder for them to handle. This relationship is similar to a game of rock-paper-scissors. In this case, the classes are not balanced with each other, but the system as a whole is in a state of balance.

Our development for automated balancing of game parameters on a prototype game model [28] has shown that Machinations balance diagrams can be used not only for manual balancing of game parameters, but also for automated balancing of game parameters, thus reducing the time spent on this lengthy process. However, although the Machination project (see Fig. 8) is still in beta testing and is still developing, the development team is open to dialogue. In addition to everything, one of the directions of development of the parameter balancing approach for the game prototype is to add various algorithms for parameter adjustment, including those based on machine learning and DPBM algorithms, automatically determining the influence of a particular parameter on the result.

3. Conclusion

The presented work shows a simple but effective visualization of the branched structure of video game storylines and presents options for automatic generation of sequels for story branches, allowing to increase the replay ability of the final product.

This solution will be another part of one big tool designed to simplify the work of game writers and improve the quality of the game narrative. After combining the functionality into a common information processing and visualization pipeline, one can hope to create a full-fledged tool for narrative prototyping. In general, a comprehensive tool should be a set of editors for various aspects of the game project.

[2] Deep Player Behavior Modeling (DPBM) is an approach that implements individual generative modeling of a player by evaluating an elementary player action in a state architecture and establishing the relationship between them using machine learning.

Immediate future plans: development of functionality for creating and tracking quests; implementation of support for more complex and massive branching structures to refine the game scenario; prototyping quantitative parameters through automatic balancing in Machination [32].

4. Acknowledgements

This paper has been supported by the Kazan Federal University Strategic Academic Leadership Program ("PRIORITY-2030").

5. References

[1] I. A. Sedyh, Industriya komp'yuternyh igr. Nacional'nyĭ issledovatel'skiĭ universitet Vysshaya shkola ekonomiki. 2020. 74 s.
[2] M. O. Riedl, V. Bulitko, Interactive Narrative: An Intelligent Systems Approach. AI Magazine. 2013. V. 34. 67 p.
[3] G. F. Sahibgareeva, V. V. Kugurakova, Koncept instrumenta avtomaticheskogo sozdaniya scenarnogo prototipa komp'yuternoj igry. Elektronnye biblioteki. 2018. T. 21. № 3-4. S. 235–249.
[4] G. F. Sahibgareeva, O. A. Bedrin, V. V. Kugurakova, Razrabotka komponenta generacii vizualizacii scenarnogo prototipa videoigr. Nauchnyj servis v seti Internet: trudy XXII Vserossijskoj nauchnoj konferencii. 2020. S. 581–603.
[5] Twine. URL: https://twinery.org/, last accessed 2021/10/21.
[6] Articy:draft. URL: https://www.articy.com/en/, last accessed 2021/10/21.
[7] Fungus. URL: https://fungusgames.com/, last accessed 2021/10/21.
[8] Storybricks Engine.
URL: https://www.youtube.com/watch?v=id-3sUo_DFU&ab_channel=Storybricks, last accessed 2021/10/21.
[9] D. Cage, Twitter blog. URL: https://twitter.com/David__Cage/status/1034374760392794112, last accessed 2021/10/21.
[10] Detroit: Become Human. URL: http://www.quanticdream.com/en#!/en/category/detroit, last accessed 2021/10/21.
[11] K. Padia, K. Bandara, C. Healey, A system for generating storyline visualizations using hierarchical task network planning. Computers & Graphics. 2019. P. 64–75.
[12] Sankey Diagram. URL: https://observablehq.com/@d3/sankey-diagram, last accessed 2021/10/21.
[13] G. F. Sahibgareeva, Primenimost' razvetvlennyh struktur dlya generacii scenarnyh prototipov videoigr. 65-ya Mezhdunarodnaya nauchnaya konferenciya Astrahanskogo gosudarstvennogo tekhnicheskogo universiteta. 2021.
[14] G. F. Sahibgareeva, O. A. Bedrin, V. V. Kugurakova, Raskadrovka kak odno iz predstavlenij scenarnogo prototipa komp'yuternyh igr. Elektronnye biblioteki. 2021. T. 24. №2. S. 408–444.
[15] G. F. Sahibgareeva, O. A. Bedrin, V. V. Kugurakova, Visualization Component for the Scenario Prototype Generator as a Video Game Development Tool. CEUR: Proceedings of the 22nd Conference on Scientific Services & Internet (SSI-2020). 2020. P. 267–282.
[16] V. V. Kugurakova, G. F. Sahibgareeva, A. Z. Nguen, A. M. Astaf'ev, Prostranstvennaya orientaciya ob''ektov na osnove obrabotki tekstov na estestvennom yazyke dlya generacii raskadrovok. Elektronnye biblioteki. 2020. T. 23. №6. S. 1213–1238.
[17] S. A. Vakatov, Razrabotka instrumenta variativnosti syuzheta s zapuskom prototipa v vide tekstovoj igry. Kazanskij (Privolzhskij) federal'nyj universitet. 2021. 36 s.
[18] E. S. Vakatova, Razrabotka funkcionala generacii prodolzheniya syuzheta dlya instrumenta prototipirovaniya syuzheta v komp'yuternyh igrah. Kazanskij (Privolzhskij) federal'nyj universitet. 2021. 33 s.
[19] B. I. Kayumov, Problemy vizualizacii razvetvlennyh syuzhetov komp'yuternyh igr. Kazanskij (Privolzhskij) federal'nyj universitet. 2021. 79 s.
[20] BioShock Infinite. URL: https://2k.com/en-US/game/bioshock-infinite/, last accessed 2021/10/21.
[21] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners. OpenAI Blog. 2019. V.1. 9 p.
[22] GPT-2. URL: https://openai.com/blog/better-language-models/, last accessed 2021/10/21.

[23] D. Sirlin, Balancing Multiplayer Games. 2009.
URL: http://www.sirlin.net/articles/balancing-multiplayer-games-part-1-definitions
[24] A. Becker, D. Görlich, What is Game Balancing? – An Examination of Concepts. In Paradigmplus, vol. 1, no. 1, pp. 22–41, Apr. 2020.
[25] I. Schreiber, Game Balance Concepts. 2010.
URL: https://gamebalanceconcepts.wordpress.com/2010/07/07/level-1-intro-to-game-balance/
[26] M. Moroşan, R. Poli, Automated Game Balancing in Ms PacMan and StarCraft Using Evolutionary Algorithms. 2017. pp. 377–392.
[27] V. Volz, G. Rudolph, B. Naujoks, Demonstrating the Feasibility of Automatic Game Balancing, GECCO, 2016, pp. 269–276.
[28] M. Beyer, A. Agureikin, A. Anokhin, C. Laenger, F. Nolte, J. Winterberg, M. Renka, M. Rieger, N. Pflanzl, M. Preuss, V. Volz, An integrated process for game balancing. In 2016 IEEE Conference on Computational Intelligence and Games (CIG), Sep. 2016, pp. 1–8, ISSN: 2325-4289.
[29] H. Chen, Y. Mori, I. Matsuba, Solving the balance problem of massively multiplayer online role-playing games using coevolutionary programming. Applied Soft Computing. 2014, 18. pp. 1–11.
[30] J. Pfau, A. Liapis, G. Volkmar, G. Yannakakis, R. Malaka, Dungeons & Replicants: Automated Game Balancing via Deep Player Behavior Modeling. 2020. pp. 1–8.
[31] J. Pfau, J. Smeddinck, R. Malaka, Towards Deep Player Behavior Models in MMORPGs. 2018. pp. 381–392.
[32] E. Adams, D. Joris, The Designer's Notebook: Machinations, A New Way to Design Game Mechanics. URL: https://www.gamasutra.com/view/feature/176033/the_designers_notebook, last accessed 2021/10/21.