Keiki: Towards Realistic Danmaku Generation via Sequential GANs

Ziqi Wang*†, Jialin Liu*†
* *Research Institute of Trustworthy Autonomous System*
Southern University of Science and Technology
†*Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation*
Department of Computer Science and Engineering
Southern University of Science and Technology
Shenzhen, China
11710822@mail.sustech.edu, liujl@sustech.edu.cn

Georgios N. Yannakakis‡
‡*Institute of Digital Games*
University of Malta
Msida 2080, Malta
georgios.yannakakis@um.edu.mt

***Abstract*—Search-based procedural content generation methods have recently been introduced for the autonomous creation of bullet hell games. Search-based methods, however, can hardly model patterns of danmakus—the bullet hell shooting entity—explicitly and the resulting levels often look non-realistic. In this paper, we present a novel bullet hell game platform named *Keiki*, which allows the representation of danmakus as a parametric sequence which, in turn, can model the sequential behaviours of danmakus. We employ three types of generative adversarial networks (GANs) and test *Keiki* across three metrics designed to quantify the quality of the generated danmakus. The time-series GAN and periodic spatial GAN show different yet competitive performance in terms of the evaluation metrics adopted, their deviation from human-designed danmakus, and the diversity of generated danmakus. The preliminary experimental studies presented here showcase that potential of time-series GANs for sequential content generation in games.**

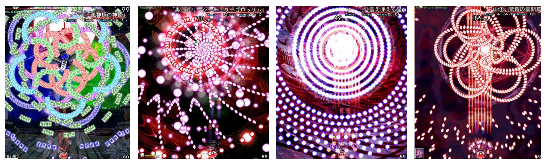
***Index Terms*—Procedural content generation, level generation, bullet hell, generative adversarial net, time-series GAN**

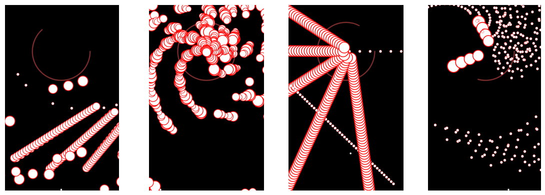
I. Introduction

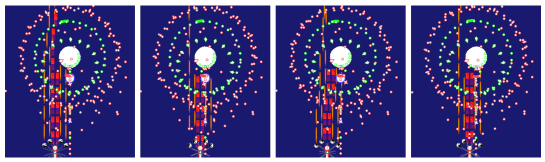
Bullet hell is a game genre in which a player survives by dodging overwhelming numbers of enemy projectiles and scores by shooting enemies down. The core component of a bullet hell game level is a "*danmaku*", which refers to the game entity (agent or opponent) that creates bullets and determines their trajectories. Danmakus in bullet hell games are typically controlled by a set of human-designed rules, aiming at the elicitation of challenging and engaging experiences for the player.

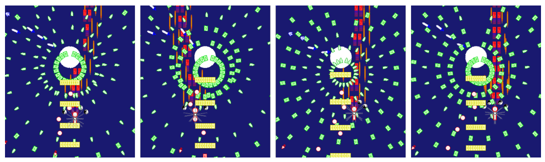
Over the last decade, a number of search-based procedural content generation (PCG) approaches have been applied widely to generate levels for platformer games, dungeon-like games and shooting games [2]–[7]; however, the generation of bullet hell levels was only been explored recently. *Talakat* [1] first applied PCG to generate bullet hell levels by searching in

This work was supported by the Guangdong Provincial Key Laboratory (Grant No. 2020B121201001), the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (Grant No. 2017ZT07X386), the Guangdong Basic and Applied Basic Research Foundation (Grant No. 2021A1515011830), the Shenzhen Science and Technology Program (Grant No. KQTD2016112514355531) and the Shenzhen Fundamental Research Program (Grant No. JCYJ20190809121403553).


(a) *Touhou Project* series (Team Shanghai Alice, since 2003).


(b) *Talakat*: Figure 6 from [1] with authors' permission.


(c) *Keiki*: danmakus generated by deep convolutional GAN.


(d) *Keiki*: danmakus generated by periodic spatial GAN.

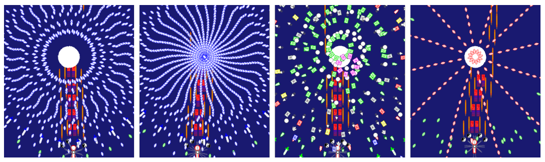

(e) *Keiki*: danmakus generated by time-series GAN.

Fig. 1. Examples of danmakus of commercial games vs. danmakus generated by *Talakat* and by our GAN generators *Keiki* (cf. Section III).

a human-designed level space represented as a grammar-based text and designed several metrics and constraints to guide the search, although the generated danmakus look rather dissimilar compared to the regular ones of commercial games (cf. Figures 1(a) and 1(b)).

Studies linked to bullet hell generation include weapon generation and rhythm game generation. In particular, Hastings *et al.* [8] evolved particle-based weapons as compositional pattern producing networks in an online manner and Hoover *et al.* [9] introduced a similar generative system that considers the background music of the game for weapon particle generation. Constrained surprise search was introduced in [10] to generate weapons in first-person shooting games. In [11], a novel neural network structure combining convolutional neural network (CNN) and recurrent neural network (RNN) was designed to generate rhythm game levels with specific degree of difficulty.

By observing the indicative outcomes of Figures 1(a) and 1(b), it appears that search-based PCG methods [2] reach a certain level of quality compared to actual bullet hell levels [1]. Through mere observation, it also seems that existing state-of-the-art approaches can hardly model the implicit danmaku patterns present in commercial-standard games. In this paper, we present *Keiki*[1] (cf. Section II) as a complementary bullet hell game platform that can represent danmakus as parametric sequences and can support neural network-based generative methods. *Keiki* currently employs three types of generative adversarial networks (GANs) [12]. The various GAN models are evaluated across three metrics designed to quantify the quality of the generated danmakus.

II. *Keiki*: Bullet Hell Platform Generator

The *Keiki* platform provides functions including danmaku design, encoding, evaluation and basic gameplay. The source code of *Keiki* is available on Github[2], including the training data and all image assets used in this work. Due to the page limit, we only describe the danmaku encoding as follows.

The design of a danmaku can be seen as specifying a tuple $\langle F, \mathbf{p} \rangle$, where F refers to the shooting rules and the vector $\mathbf{p}$ specifies the values of F's control parameters, usually defined in the construction method of the implementation class. At the t^{th} frame, the danmaku calls the implemented bullet builder several times according to its shooting rules (F) and parameter values ($\mathbf{p}$). This process is denoted by $F(t|\mathbf{p}) \mapsto (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{N^t})$, where $N^t \in \mathbb{N}$ is the number of times that this danmaku calls the bullet builder at the t^{th} frame and $\mathbf{x}_i$ is the parameter vector used at the i^{th} call. When shooting multiple times is desired (e.g., L times), $F(t|\mathbf{p})$ will be repeatedly called for $t \in \{1, 2, \cdots\}$ until $t = T$ such that $\sum_{t=0}^{T} N^t \geq L$ is satisfied. Thus, the length of the parametric sequence to be generated is L. To reduce the dimensionality and data redundancy, the data compress method proposed in the work of [13] is used, described as follows. A parameter itv (short for interval, corresponding to $time$ in [13]), representing the time passed since the last call of bullet builder, is also added to the parametric sequence.

[1]The name "*Keiki*" comes from a character in *Touhou Kikeijuu $\sim$ Wily Beast and Weakest Creature* (Team Shanghai Alice, 2019).

[2]https://github.com/PneuC/Keiki

III. Danmaku Generation

In Section III-A we outline the three GAN methods used for danmaku generation. Section III-B introduces the metrics we used to evaluate the danmaku performance. Our preliminary experimental results are presented and discussed in Section III-C.

A. Danmaku Generation with GANs

Deep convolutional GANs, periodic spatial GANs and time-series GANs are considered separately and employed to generate parametric sequence of length 64. Generators are trained on a dataset of 34 danmakus implemented by the first author of this paper. These danmakus are mainly imitations of danmakus found in *Touhou Project*[3] series.

During training, data augmentation is applied as follows. Each time a danmaku $\langle F, \mathbf{p} \rangle$ is loaded from training data, a Gaussian mutation is added to its parameters, i.e., $\mathbf{p} \leftarrow \mathbf{p} + 0.05\ \mathcal{N}(0, \mathbf{p} \times \mathbf{I})$, before feeding it into the model.

1) Time-series GAN: We implement time-series GAN (TimeGAN) [14] as an alternative PCGML approach, as it combines autoencoder and supervised loss to help learning temporal dynamics from training data. Not only a generator and a discriminator are involved in TimeGAN, but also a pair of an embedder (encoder) and a reconstructor (decoder), trained with a reconstruction loss, a supervised loss and an adversarial loss jointly. For all these models in TimeGAN, a 3-layer stacked LSTM with 128 hidden units in each layer is used and the logistic function is employed in the output layer. The dimensionality of the autoencoder's hidden space is set as 24. The noise input to TimeGAN is composed of a global noise Z^g and a periodic noise Z^p. For any $i \in \{1, 2, \ldots, T_Z\}$, the input noise is $Z^g \oplus Z^g_i$, where $\oplus$ denotes concat. Z^g is sampled once and duplicated T_Z times (T_Z is the spatial length of the noise, set as 10 here), while Z^p is sampled as $Z^p_i = sin(i \cdot K_1(Z^g) + K_2(Z^g))$, where K_1 and K_2 are two multi-layer perceptrons.

2) Periodic spatial GAN: The periodic spatial GAN [15] with sequential noise input is employed as our second, alternative, generative model. The noise input to periodic spatial GAN is the same as that of TimeGAN. We use four one-dimensional convolutions with no padding. The kernel size and stride are set as $(4, 1), (4, 2), (4, 1), (4, 2)$, respectively, for each layer. Such a design guarantees that the generator will create sequences of length 64 when the spatial length of input noise is 10, which is directly comparable with other employed GAN architectures. The structure of the discriminator is symmetric to the generator. Both the generator and the discriminator use the logistic activation function in the final layer and the ReLU activation function in all other layers.

[3]https://touhou.fandom.com/wiki/Touhou_Project

3) Deep convolutional GAN: A vanilla deep convolutional GAN (DCGAN) [16] is implemented as a non-sequential baseline. We used 5 one-dimensional transposed convolutional layers in the generator. The kernel size, stride and padding of the first and the remainder layers are $(4, 1, 0)$ and $(4, 2, 1)$, respectively. The number of output channels of the first layer is 256 and are reduced by 2 times at each layer that follows. The structure of the discriminator is also symmetric to the generator, and the activation function is the same as the one used in the periodic spatial GAN.

B. Evaluation Metrics

Three feature-based metrics are designed to evaluate a generated danmakus D. The first one is *shooting frequency*, SF, which determines whether all bullets are shot in a very short period or progressively.

$$SF(D) = L/T, \tag{1}$$

where L is the number of shooting times (i.e., the length of parametric sequence) and T is danmaku's duration.

The second metric used is *mean momentum*, defined as the sum of all bullets on the screen over time. Let B^t be the set of all the bullets at the t^{th} frame. For any bullet b, b_w and b_s denote respectively its weight (a scalar value proportional to the size of its image) and its speed. The mean momentum of danmaku D is calculated as:

$$MM(D) = \frac{1}{T}\sum_{i=1}^{T}\sum_{b\in B^t} b_w b_s, \tag{2}$$

where T is the danmaku's duration.

Coverage is another metric used to estimate the degree of game difficulty. The game screen is split into grids of $8 * 8$ pixels and the percentage of regions that is covered by any bullet at any frame is computed. Let r and c be the number of rows and columns of the regions, respectively. $cov^t_{i,j}$ determines whether the region at position i, j is covered by at least one bullet at t^{th} frame. The coverage metrics is as follows:.

$$C(D) = \frac{1}{rc}\sum_{i=1}^{r}\sum_{j=1}^{c}\mathbf{1}(\exists\ t \in [1, T], cov^t_{i,j}), \tag{3}$$

where $\mathbf{1}(p)$ is 1 if the proposition p holds, otherwise 0. Higher C values imply more difficult levels.

Although the aforementioned metrics may not suffice to determine if generated danmakus are of good quality, they can be used to quantify the similarity between real and generated danmakus with distance measures, such as those of the KL-divergence family.

C. Experimental Study and Discussion

After preliminary hyper-parameter tuning, the following setting is used in our experiments. The batch size is set to 12 for all GANs. The learning rate is $2 * 10^{-4}$ for the DCGAN and the periodic spatial GAN, while it is set to $2 * 10^{-3}$ for the TimeGAN. We optimise the DCGAN and the periodic spatial GAN via Adam and the TimeGAN via RMSprop. The DCGAN and periodic spatial GAN are trained separately for $5,000$ iterations. The autoencoder of TimeGAN is pre-trained for $5,000$ iterations, then the generator is trained with supervised loss only for 500 iterations; finally we the trained all the models jointly for $5,000$ iterations.

Figure 2 shows how the values of metrics introduced in Section III-B change during training. The DCGAN performs the worst since its metric values are the ones furthest to the ones from real data. The periodic spatial GAN, on the other hand, shows more stable performance compared to that of the TimeGAN. The instability of TimeGAN in our experiments may be explained by its complex architecture which, in turn, calls for larger amounts of training data. According to Figure 2, however, the TimeGAN approach has the highest standard derivation across all metrics, which implies the highest diversity. The other GANs appear to suffer from mode collapse, i.e., they only generate data of one or a limited number of patterns.

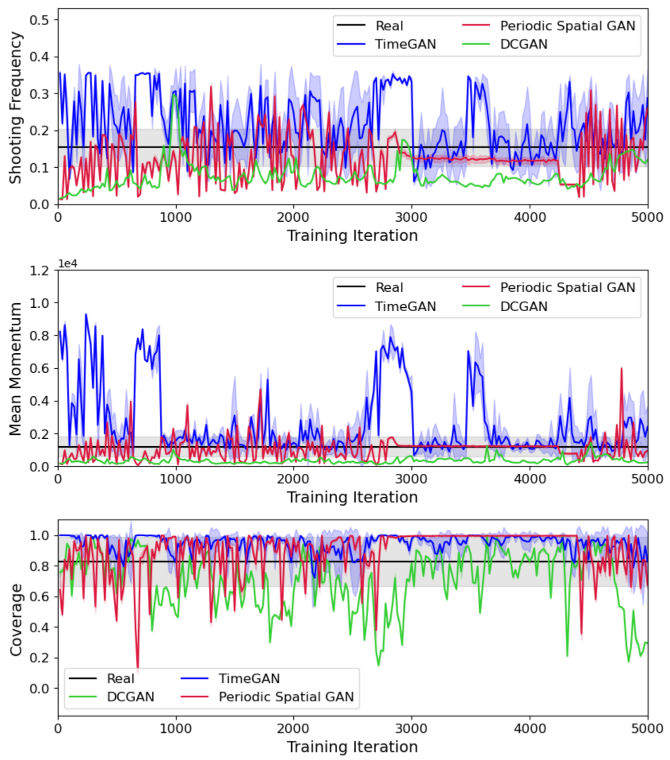


Fig. 2. Comparison of GANs on shooting frequency (top), mean momentum (middle) and coverage (bottom) during training. Every 20 iterations, the trained GAN generates 30 samples for evaluation. Solid curves depict the mean value over 30 samples and shadow areas illustrate the standard derivation.

Figures 1(c), 1(d) and 1(e) show indicative examples of danmakus generated by the different GANs. DCGAN generates almost similar danmakus; periodic spatial GAN generates danmakus of similar patterns whereas TimeGAN appears to generate far more diverse danmakus compared to the other two GANs.

IV. Further Discussion

Generating danmakus via PCGML [6] using GANs is a challenging task; especially when training data is not readily available. Bootstrap methods [17] or PCG via reinforcement learning (PCGRL) [18], [19] are worth investigating to overcome this challenge, but collecting more human-designed danmakus as training data is also important. While implementing and coding danmakus is time-consuming, videos of danmakus are easier to obtain from real games; thus, learning to represent danmakus directly from videos appears to be a promising future direction. The design of alternative GAN architectures is also important to be investigated for future work.

Another important direction for future research is agent-based evaluation. An A^* agent has already been implemented in *Keiki*. Agent-based testing, however, is currently inefficient due to the slow simulation times. In bullet hell games hundreds of bullets may exist simultaneously at each frame; in *Keiki*, bullets' moving directions can vary depending on player's current or previous positions, which makes the real-time simulation CPU intensive. If more efficient agents are implemented or the game simulation can eventually be accelerated, we plan to add constraints regarding the playability of generated danmakus by using techniques such as constrained adversarial nets [20].

The three metrics currently used this work are intuitively designed. Designing new evaluation metrics and employing other PCG methods, such as that in [1], as baselines of the *Keiki* platform define top priorities for future work.

V. Conclusion

In this paper, we presented *Keiki*, a novel bullet hell platform which allows encoding danmakus into a parametric sequence so that various artificial level generators can be used to generate danmakus. We also introduce GANs [12] as an alternative PCGML [6] mechanism that learns to represent human-designed danmakus—and hence generates more realistic content (cf. Section III)—and employ three different GAN architectures to generate them. Experimental results on *Keiki* show that both the TimeGAN and the periodic spatial GAN methods have the potential of generating realistic danmakus. The TimeGAN, in particular, generated more diverse danmakus, while the periodic spatial GAN is not sensitive to the length of parametric sequence since it is based on CNNs. Danmaku generation with GANs can be further enhanced though agent-based evaluation and constrained optimisation to ensure playability and yield more realistic danmakus that feature particular human-designed patterns or features.

References

[1] A. Khalifa, S. Lee, A. Nealen, and J. Togelius, "Talakat: Bullet hell generation through constrained map-elites," in *Proceedings of The Genetic and Evolutionary Computation Conference*, 2018, pp. 1047–1054.

[2] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based procedural content generation: A taxonomy and survey," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 172–186, 2011.

[3] J. Togelius, A. J. Champandard, P. L. Lanzi, M. Mateas, A. Paiva, M. Preuss, and K. O. Stanley, "Procedural content generation: Goals, challenges and actionable steps," in *Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik*, 2013.

[4] N. Shaker, J. Togelius, and M. J. Nelson, *Procedural Content Generation in Games*. Springer, 2016.

[5] S. Risi and J. Togelius, "Increasing generality in machine learning through procedural content generation," *Nature Machine Intelligence*, vol. 2, no. 8, pp. 428–436, 2020.

[6] A. Summerville, S. Snodgrass, M. Guzdial, C. Holmgård, A. K. Hoover, A. Isaksen, A. Nealen, and J. Togelius, "Procedural content generation via machine learning (PCGML)," *IEEE Transactions on Games*, vol. 10, no. 3, pp. 257–270, 2018.

[7] J. Liu, S. Snodgrass, A. Khalifa, S. Risi, G. N. Yannakakis, and J. Togelius, "Deep learning for procedural content generation," *Neural Computing and Applications*, vol. 33, pp. 19–37, 2021.

[8] E. J. Hastings, R. K. Guha, and K. O. Stanley, "Evolving content in the galactic arms race video game," in *2009 IEEE Symposium on Computational Intelligence and Games*. IEEE, 2009, pp. 241–248.

[9] A. K. Hoover, W. Cachia, A. Liapis, and G. N. Yannakakis, "Audioinspace: Exploring the creative fusion of generative audio, visuals and gameplay," in *International Conference on Evolutionary and Biologically Inspired Music and Art*. Springer, 2015, pp. 101–112.

[10] D. Gravina, A. Liapis, and G. N. Yannakakis, "Constrained surprise search for content generation," in *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2016, pp. 1–8.

[11] Y. Liang, W. Li, and K. Ikeda, "Procedural content generation of rhythm games using deep learning methods," in *Joint International Conference on Entertainment Computing and Serious Games*. Springer, 2019, pp. 134–145.

[12] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14. Cambridge, MA, USA: MIT Press, 2014, p. 2672–2680.

[13] O. Mogren, "C-RNN-GAN: Continuous recurrent neural networks with adversarial training," *arXiv preprint arXiv:1611.09904*, 2016.

[14] J. Yoon, D. Jarrett, and M. van der Schaar, "Time-series generative adversarial networks," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.

[15] U. Bergmann, N. Jetchev, and R. Vollgraf, "Learning texture manifolds with the periodic spatial GAN," in *ICML*, 2017.

[16] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[17] R. R. Torrado, A. Khalifa, M. C. Green, N. Justesen, S. Risi, and J. Togelius, "Bootstrapping conditional gans for video game level generation," in *2020 IEEE Conference on Games (CoG)*. IEEE, 2020, pp. 41–48.

[18] A. Khalifa, P. Bontrager, S. Earle, and J. Togelius, "PCGRL: Procedural content generation via reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 16, no. 1, 2020, pp. 95–101.

[19] T. Shu, J. Liu, and G. N. Yannakakis, "Experience-driven PCG via reinforcement learning: A Super Mario Bros study," in *The 2021 IEEE Conference on Games (CoG)*, 2021, p. accepted.

[20] L. Di Liello, P. Ardino, J. Gobbi, P. Morettin, S. Teso, and A. Passerini, "Efficient generation of structured objects with constrained adversarial networks," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 14 663–14 674.