Computer Systems Science & Engineering
DOI:10.32604/csse.2021.016377
Article

Tech Science Press

A Fire Escape Simulation System Based on the Dijkstra Algorithm

Haolong Yang[1], Chunqiang Hu[1], Guwei Li[2,*] and Jingchun Fan[3]

[1]ChongQing University, ChongQing, 400000, China
[2]Zhejiang Dongfang Polytechnic, WenZhou, 325000, China
[3]Compugen Ltd, Toronto, M2J 4A6, Canada
*Corresponding Author: Guwei Li. Email: 3168897@qq.com
Received: 31 December 2020; Accepted: 13 April 2021

Abstract: Despite the support of all kinds of fire prevention measures and high-tech fire prevention equipment, fires still occur frequently because of both anthropogenic factors and natural disasters. This issue has drawn the attention of schools, all levels of government, and other organizations. Many types of organizations carry out fire drills throughout the year. Because this kind of drill cannot anticipate the specific circumstances of each fire, which are generally far more complicated than drills, most people cannot correctly choose the optimal escape route from real fires. Thus, a fire-scene virtual simulation system based on the Dijkstra algorithm is here proposed to address such problems as casualties caused by frequent fires and the inability of most people to correctly choose a fire escape route. This virtual fire escape simulation system uses Maya to carry out 3D reconstruction of the fire scene, the Unity engine to conduct interactive function design, and the Dijkstra algorithm to calculate the best escape route. The results of the example indicate that the simulation system solves the problems of the traditional simulation system, such as stiffness, lack of intelligence, and poor simulation.

Keywords: Dijkstra algorithm; virtual fire escape simulation system; fire

1 Introduction

Fires occur frequently, and this has drawn the attention of schools, all levels of government, and many other types of organizations [1]. Most organizations that work out of modern buildings hold on-site fire drills [2–6], hoping to reduce casualties from fire through simulation exercises [7]. However, this kind of drill cannot fully anticipate the specific circumstances of each fire. A real fire situation is complex, and ordinary people often cannot choose the optimal escape route in the face of confusion. Traditional fire simulation systems cannot provide an escape route selection function, meaning that the fire safety simulation exercise cannot achieve its intended purpose.

A fire scene virtual simulation system [8,9] based on the Dijkstra algorithm [10] is here proposed to solve these problems. This virtual [11] fire escape simulation system uses Maya to carry out a three-dimensions (3D) reconstruction [12,13] of the fire scene, the Unity engine to conduct interactive function design, and the Dijkstra algorithm to calculate the best escape route. The results of an example using the

Dijkstra algorithm indicate that the simulation system solves the problems of the traditional simulation bsystem, such as rigidity, unintelligence, and poor imitation.

2 The Dijkstra Algorithm

The Dijkstra algorithm, a representative shortest route algorithm [14,15], is mainly used to determine the shortest route between points (between nodes). The core feature is that one of the nodes is taken as the center node and spreads out layer by layer until the target node is obtained as follows:

(1) Specify a node. For example, obtain the shortest route between node "A" and other nodes (Fig. 1).

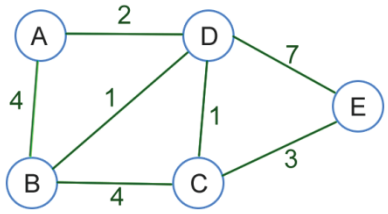

Figure 1: Node diagram

(2) Create two collections (L, Y), where collection L is used to store the nodes that have obtained the shortest route and collection Y is used to store nodes that have not obtained the shortest route.

(3) Initialize collections L and Y. At the beginning, the collection L contains: A->A=0, Y; the collection Y contains: A->B=4, A->C=∞, A->D=2 and A->E=∞ (Fig. 2).

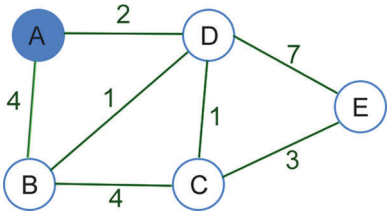

Figure 2: Select node A

(4) Obtain the node with the shortest route in the collection Y and add it to collection L. For example, A->D=2. If the distance from AD plus D to B, C, and E is less than the distance from A to B, C, and E, collection Y will be updated. At this point, collection L contains: A->A=0, A->D=2; collection Y contains: A->B=3, A->C=3, and A->E=9 (Fig. 3).

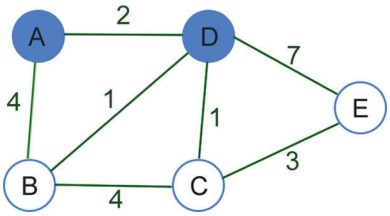

Figure 3: First update of Y

(5) At this point, A->B and A->C are already the shortest route 3, so collection Y is updated. Collection L contains: A->A=0, A->D=2, A->B=3 and A->C=3; and collection Y contains: A->E=9 (Fig. 4).

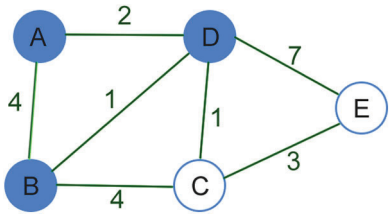

Figure 4: Second update of Y

(6) Judge the distance from A to D to E and the distance from A to C to E. Take the shortest distance, and then update collection Y. At this point, collection L contains: A->A=0, A->D=2, A->B=3, A->C=3, and A->E=6. Collection Y contains: null (Fig. 5).

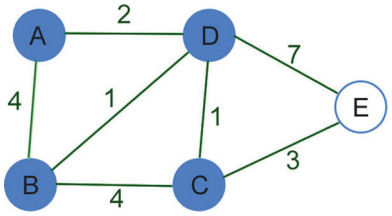

Figure 5: Third update of Y

(7) The shortest route between A and each other node is finally obtained.

3 Fire Escape Simulation System Based on the Dijkstra Algorithm

The fire escape simulation system based on the Dijkstra algorithm uses Maya to perform 3D reconstruction of the fire scene, the Unity engine to conduct interactive function design, and the Dijkstra algorithm to calculate the best escape route, in cooperation with technology such as the Visual Effect Graph and Shader Graph, to carry out special effect simulations of many fire scenarios. The system will be released on several platforms, principally the Virtual Reality (VR) platform and the mobile platform. Students can use VR headsets and mobile phones to experience realistic fire simulations in an interactive manner, which can give a more grounded sense of then experience while maintaining the safety of a fire drill.

3.1 Fire Escape Simulation System Development Process

The methods used to develop the fire escape simulation system based on the Dijkstra algorithm and the overall process of the proposed method are shown in Fig. 6.

The escape function module based on the Dijkstra algorithm was developed in the Unity engine. At the same time, intersection of ball detection technology with the addition of explosion point technology was used so that fire and explosion effects could be added for each explosion point. An eye-catching user interface (UI) was designed to provide the users with a clear sense of experience. Finally, the system generated was published to HTC VIVE.

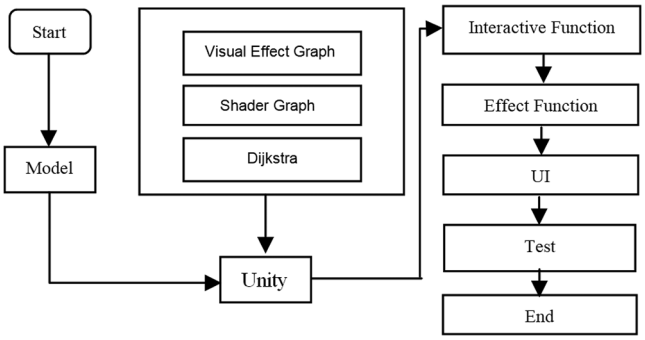

Figure 6: Development of the overall process of the fire escape simulation system

3.2 Maya Technology

Produced by Autodesk, Maya [16–18] is a powerful 3D animation software that is often applied to 3D modeling, film and television animation, and film and television special effects. The most commonly used modeling method is polygon modeling, where developers can create clay figures to precisely control each point of the model. The operation mode of modeling is novel, and the Unique Visitor (UV) module in the new version of the software has been upgraded, which is convenient and suitable for VR application. For this reason, Maya was chosen in the study for modeling, UV processing, and map production of the dormitory scene.

3.3 Unity Technology

This fire escape simulation system has very high criteria for the authenticity of 3D scenes, which requires an engine capable of producing 3A level images. Many interactive functions and the production of UI should be added, and the implementation of these functions and effects can be achieved only by Unity.

Unity [19–21] is an engine for developing 3D games, especially in the mobile game market. The development language is C sharp (C#), which is based on module-oriented development and is compatible with all platforms. Unity is used to achieve the special effects display and interactive functions in the system.

Unity development involves all module-oriented development, and the Visual Effect Graph [22] module is mainly used in the system. The powerful effect and control power of this module were key to our realization of vivid special effects. However, the Shader Graph module was also necessary for us to realize physical rendering.

4 Visual Effect Graph Technology

There are many explosion and flame effects in the fire simulation system, but these effects, especially photographic-level effects, are difficult to achieve using traditional computer graphics (CG) technology [23,24]. Moreover, operation efficiency should be considered. The use of the latest Visual Effect Graph technology of Unity ensures smooth operation of the system and helps to achieve photographic-level effects.

Visual Effect Graph is a new technology that was released after Unity 2018.3, which is a new special effects tool. It is similar to the original particle system, but its most intuitive feature is a visual node that supports programming and runs beyond the Graphics Processing Unit (GPU) [25,26]. These features enable it to support a larger graphic computation burden and provide a more flexible development space for developers and creators. Therefore, it can be used to create many special effects in AAA games.

The methods of using Visual Effect Graph technology in Unity are shown as follows.

1) Installation and configuration

Visual Effect Graph needs to be installed by the Package Manager. It currently only supports the High-Definition Render Pipeline (HDRP), so HDRP needs to be installed. After installation, the HDRP configuration file should be created in the create menu of the project window.

2) Creation and editing of VFX Graph

First, a Visual Special Effects (VFX) Graph asset must be created. After dragging it into the scene, a game object is generated automatically. A Visual Effect module will be automatically added to Unity. Refer to the VFX Graph asset created before, which can be edited after being opened (Fig. 7).

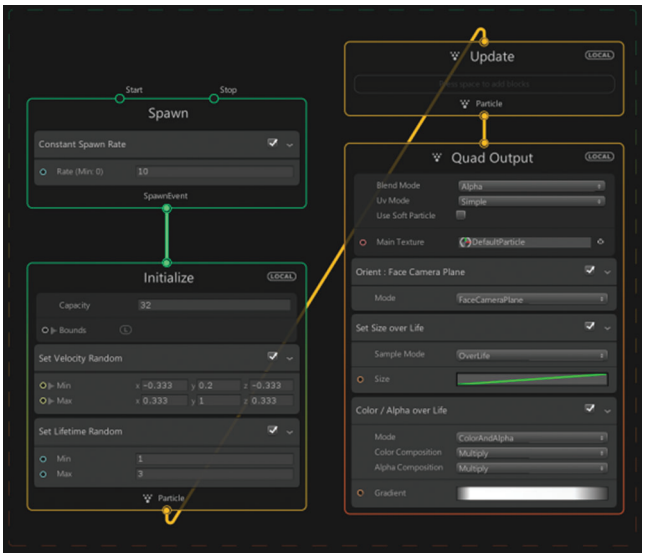

Figure 7: VFX graph editing interface

System: The collection of several contexts enclosed by dashed lines is called the System. One System contains Initialize, Update, and Output. One VFX Graph can contain more than one System.

Spawn: This context defines the quantity and time of particles that are generated.

Initialize: The context, similar to the start method of writing a script, is used to initialize the particles and should be started with the Capacity and Bounds.

Capacity defines the maximum quantity of particles that can exist at one time. This value is important because it determines how much memory is allocated initially. The value should be set according to the quantity of particles produced. It can commonly be calculated by the formula: Rate × Max Lifetime = Capacity.

Bounds: Defines the simulation region of the particles.

Update: The context, similar to the Update method of writing a script, is used to set how the particle changes over time. Forces such as collision and force fields can be used by it.

Output: Used to render particles. It determines the type, texture, color, and orientation of the generated particles. A System can contain more than one Output.

3) Operational test

By setting the appropriate parameter values, the effect shown in Fig. 8 can be obtained.

Figure 8: Simple flame effect test

5 Shader Graph Technology

1) Creation and setting

To use the Shader Graph, we installed the Shader Graph package and the Universal RP package (the previous name of Unity 2019.3 is Light Weight RP). Create and configure the programmable rendering pipeline (SRP), and then create a new Physically Based Rendering (PBR) Graph. By opening the PBR Graph file, you can start to set the material effect based on physical rendering.

2) PBR rendering effect test

Through the connection of various nodes and debugging parameters, the edge luminous effect shown in Fig. 9 can be obtained.

Figure 9: Explosion effect test

6 Overall Design of the System

The escape route module was improved and added to the existing VR reproduction system owing to its functional modules and shortcomings.

The function modules are as follows:

1) Scene module: Create fire scene models, add special effects in the process of various fires, and add a real-time updated UI.

2) VR core module: The Dijkstra algorithm, the VFX Graph technology, and the Shader Graph technology.

3) Real-time simulation: Triggering events and real-time display of interactive effects.

7 Case Study

The Fire and Mars effects were added to the scene by the Visual Effect Graph technology, and all material rendering was realized based on the Shader Graph technology. A real-time UI for real-time interaction and escape route function module based on the Dijkstra algorithm was added to select the optimal route. According to the research, this system enabled users to use VR to control the scene and UI interaction, and truly restored various effects of the fire scene.

8 Conclusion

A fire-scene virtual simulation system based on the Dijkstra algorithm was developed to solve the problems of traditional simulation systems, such as casualties caused by frequent fires and the inability of most people to correctly choose a fire escape route. This paper discusses the overall design, development process, and technical solution plan of the fire escape simulation system. The key technologies such as the Dijkstra algorithm, Visual Effect Graph, and Shader Graph and explain how various explosions and flame special effects simulations are realized and how the fire escape simulation prototype system was developed. The results of the case study indicate that this method can reproduce visual effects of the fire scene, solving the problems of the traditional fire simulation system, which include the lack of any optimal escape route module, lack of intelligence, and low level of realism.

Acknowledgement: We would like to thank LetPub (www.letpub.com) for providing linguistic assistance during the preparation of this manuscript.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest regarding the present study.

References

[1] K. Chang, T. Chen, Y. Lee, Y. Lin and T. Nguyen, "Study of the high-tech process mechanical integrity and electrical safety," in *2019 14th Int. Microsystems, Packaging, Assembly and Circuits Technology Conf. (IMPACT)*, Taipei, Taiwan, pp. 162–165, 2019.

[2] W. Fang, F. Zhang, Y. Ding and J. Sheng, "A new sequential image prediction method based on lstm and dcgan," *Computers, Materials & Continua*, vol. 64, no. 1, pp. 217–231, 2020.

[3] W. Fang, L. Pang and W. N. Yi, "Survey on the application of deep reinforcement learning in image processing," *Journal on Artificial Intelligence*, vol. 2, no. 1, pp. 39–58, 2020.

[4] A. Elbir, H. O. Ilhan and N. Aydin, "The implementation of optimization methods for contrast enhancement," *Computer Systems Science and Engineering*, vol. 34, no. 2, pp. 101–107, 2019.

[5] R. Zhang and X. Zhao, "The application of folk art with virtual reality technology in visual communication," *Intelligent Automation & Soft Computing*, vol. 26, no. 4, pp. 783–793, 2020.

[6] L. Li, Y. Wei, L. Zhang and X. Wang, "Efficient virtual resource allocation in mobile edge networks based on machine learning," *Journal of Cyber Security*, vol. 2, no. 3, pp. 141–150, 2020.

[7] J. Wu, J. Guo and G. Liu, "The planning method of fire-using scheme based on ordinal optimization theory," in *2015 Chinese Automation Congress (CAC)*. Wuhan, China, 216–219, 2015.

[8] A. Altalbe, "Performance impact of simulation-based virtual laboratory on engineering students: A case study of Australia virtual system," *IEEE Access*, vol. 7, pp. 177387–177396, 2019.

[9] W. Zhang, X. Chen and J. Jiang, "A multi-objective optimization method of initial virtual machine fault-tolerant placement for star topological data centers of cloud systems," *Tsinghua Science and Technology*, vol. 26, no. 1, pp. 95–111, 2021.

[10] N. Jasika, N. Alispahic, A. Elma, K. Ilvana, L. Elma *et al.,* "Dijkstra's shortest path algorithm serial and parallel execution performance analysis," in *2012 Proc. of the 35th Int. Convention MIPRO*, Opatija, pp. 1811–1815, 2012.

[11] C. Zhao, T. Wang and A. Yang, "A heterogeneous virtual machines resource allocation scheme in slices architecture of 5g edge datacenter," *Computers, Materials & Continua*, vol. 61, no. 1, pp. 423–437, 2019.

[12] Y. Ma, Y. Wang, X. Mei, C. Liu, X. Dai *et al.,* "Visible/infrared combined 3D reconstruction scheme based on nonrigid registration of multi-modality images with mixed features," *IEEE Access*, vol. 7, pp. 19199–19211, 2019.

[13] Y. Altmann, S. McLaughlin and M. E. Davies, "Fast online 3D oeconstruction of dynamic scenes from individual single-photon detection events," *IEEE Transactions on Image Processing*, vol. 29, pp. 2666–2675, 2020.

[14] Y. Gao, "An improved shortest route algorithm in vehicle navigation system," in *2010 3rd Int. Conf. on Advanced Computer Theory and Engineering (ICACTE)*, Chengdu, China, pp. 363–366, 2010.

[15] Chang Wook Ahn and R. S. Ramakrishna, "A genetic algorithm for shortest path routing problem and the sizing of populations," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 6, pp. 566–579, 2002.

[16] S. Xia, "Application of Maya in film 3D animation design," in *2011 3rd Int. Conf. on Computer Research and Development*, Shanghai, China, pp. 357–360, 2011.

[17] G. Can, J. Odobez and D. Gatica-Perez, "Maya codical glyph segmentation: A crowdsourcing approach," *IEEE Transactions on Multimedia*, vol. 20, no. 3, pp. 711–725, 2018.

[18] J. Kinsman and D. Asher, "Orbital dynamics of highly probable but rare Orionid outbursts possibly observed by the ancient Maya," *Monthly Notices of the Royal Astronomical Society*, vol. 493, no. 1, pp. 551–558, 2020.

[19] S. Pei and K. Chang, "Odd ramanujan sums of complex roots of unity," *IEEE Signal Processing Letters*, vol. 14, no. 1, pp. 20–23, 2007.

[20] D. Jakelić and A. A. de Moura, "Tensor products, characters, and blocks of finite-dimensional representations of quantum affine algebras at roots of unity," *International Mathematics Research Notices*, vol. 2011, no. 18, pp. 4147–4199, 2011.

[21] M. Zhu, "Regular representations of quantum groups at roots of unity," *International Mathematics Research Notices*, vol. 2010, no. 15, pp. 3039–3065, 2010.

[22] B. Schroeder, S. Tripathi and H. Tang, "Triplet-aware scene graph embeddings," in *2019 IEEE/CVF Int. Conf. on Computer Vision Workshop (ICCVW)*, Seoul, Korea (South), pp. 1783–1787, 2019.

[23] F. Peng, L. Yin, L. Zhang and M. Long, "CGR-GAN: CG facial image regeneration for antiforensics based on generative adversarial network," *IEEE Transactions on Multimedia*, vol. 22, no. 10, pp. 2511–2525, 2020.

[24] W. Lee and S. Hong, "28 GHz RF front-end structure using CG LNA as a switch," *IEEE Microwave and Wireless Components Letters*, vol. 30, no. 1, pp. 94–97, 2020.

[25] G. Vigueras and J. M. Orduña, "On the use of GPU for accelerating communication-aware mapping techniques," *Computer Journal*, vol. 59, no. 6, pp. 836–847, 2016.

[26] S. Keskin and T. Kocak, "GPU-based gigabit LDPC decoder," *IEEE Communications Letters*, vol. 21, no. 8, pp. 1703–1706, 2017.