International Journal of Engineering & Technology, 7 (4.40) (2018) 57-60

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Collaborative 3D Terrain Editing Application

Mahyuddin K.M. Nasution[1], Jos Timanta Tarigan[1*], Ivan Jaya[1], Sri Melvani Hardi[1], Syahriol Sitorus[2]

[1]Faculty of Computer Science and Information Technology, Universitas Sumatera Utara
[2]Faculty of Mathematics and Natural Science, Universitas Sumatera Utara
**Corresponding author E-mail:jostarigan@usu.ac.id*

Abstract

In 3D content creation, developing a terrain may be time consuming due to most of current 3D applications require a vast terrain. In this paper, we introduce a collaborative terrain editor that allows multiple users to collaborate in real time. The objective of the application is to increase the productivity in developing a vast terrain in 3D environment. The application is based on a client server architecture where each editor applications are connected to a single server that collects and distributes the editing process done by each user. We perform our test in a local area network environment with multiple clients connected to a server and observe the performance and the usability of the software. The test results show the system is capable to perform real-time collaborative terrain editing with insignificant delay. Moreover, most of our users agree that using the system may increase the speed in 3D terrain creation.

Keywords: *Collaborative Work, Computer Graphic, Client-Server Systems*

1. Introduction

In most 3D application, creating terrain can be difficult. Some applications require massive terrain content while others require a certain level of quality. Certain games categories, such as flight simulators, allow its users to navigate quickly through the environment. Thus, developing a massive and non-repetitive terrain content is crucial to create a believable environment. Other games such as first-person shooter highly depends on the quality of the terrain since it may affect the gameplay.

A few solutions have been developed to decrease the effort to create 3D terrain. Procedural generated terrain is one of the most common way to create terrain in short time. There are numerous methods that are capable to create a high-quality terrain such as noise-based, hydraulic erosion [1]–[3], and thermal displacement [4]. While these methods are commonly used in creative industry, generating terrain procedurally is not always an optimal solution due to lack of user's control [5]. Other approaches focus on enhancing the tools on developing the terrain. The implementation of interactive sketching [6], heightfield brush [7] and reference image [8] are some of the methods used to increase user's control. Commercial terrain authoring application such as Terragen [9] and Gaia [10] combine these methods to enhance user's productivity in developing a realistic terrain.

In this paper, we propose a new method that may increase user's productivity in developing 3D terrain. The main idea of the proposed system is to allow multiple users to collaborate in real time to create a terrain. To present this idea, we developed a terrain editor with real-time collaboration feature. We created a Unity based client terrain editing application which connected to a single server. Users are able to perform terrain editing through the client application and the server will merge the changes and distributed the result to each user. During the test, we observe the usability of the system by measuring the delay between user's input and changes made by the system.

2. Page layout

The term Computer Supported Cooperative Work (CSCW) was first coined in 1984 by Iren Greif of MIT and Paul Cashman of Digital Equipment Corporation [11]. The idea of CSCW was triggered by the increasing trend of office automation which, most of the times requires, group work. Reinhard et al. outlined several concepts and architectures for CSCW applications/systems [12].

Developing a system to create an effective and efficient Computer Supported Cooperative Work (CSCW) is not a trivia task. Most of the works focused on users comfort interacting with the system which includes developing user-friendly software, addressing social dynamics of group activities, standardizing various terms, and handling interactions between multiple users [13]. Other than the capability of current technology, there are many factors that need to be considered to enhance the roll of CSCW in a company or organization. As outlined by Dourish et al. [14], users' awareness and coordination are two of the most important factors in cooperative work. The paper proposed three concepts of coordination that can be applied to increase awareness amongst cooperative users; explicit approach through directed messaging, strong notion of roles and activities of each users, and shared feedback. The latest is argued as the best method to increase users' awareness yet implementing this method requires the most effort. A work from Grudin [15] describes the cause of many implementation of CSCW that has fallen far short of expectation. He listed the three main contributing factors that lead to these failures: disparity between those who will benefit from the application and those who must do additional work to support it, ill-fated development effort, and the failure to learn from experience as an evaluation for the application.

Despite all the problems and obstacles in developing and implementing it, previous works and researches suggested that current CSCW method has been proven capable to aid multi-user task such as remote medical diagnosis support and teleconsultation

[16], improving orthography system development process [17], and N-Screen Mobile Advertising [18]. Other researches investigated the use of various display devices to implement CSCW such as tabletop display [19], stereoscopic display [20], and virtual reality system [21]. Several studies have also been done in implementing CSCW in multimedia. Imae et al. proposed a conflict-free data structured called ChainVoxel. The solution guarantees the eventual consistency on the shared data when multiple users. Similarly, Ha et al. developed a collaborative 3D editing tool based on cloud storage, Lets3D [22]. The system allows users to collaborate in real time by using a web-based 3D content authoring application.

3. System Requirement

In this section, we will go through the design of our proposed system. We will go through the requirement of the system and build the application architecture to meet the requirements. The architecture of our current system follows our previous work [23]. Our final objective is to create collaborative multi-user multi-role terrain rendering application. The figure below is the basic architecture of the Collaborative 3D Terrain Editing Application.

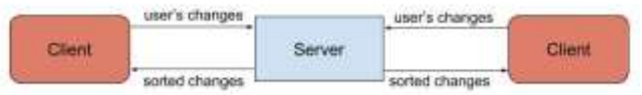

Fig. 1: Architecture for Collaborative 3D Terrain Editing Application

The main objective of the system is to allow multiple client to collaborate on a single terrain map. Hence, the main requirement of the system is to be able to be run concurrently in multiple devices while connected to a single server. The system must also be able to render a terrain in 3D format. In the current version, we define a terrain as a set of polygons with certain elevation/height value. Additionally, the system must be able to show changes based on its source. This is important to help user understand which changes is made by another client. Hence, we apply a color-coded polygon based where each client has a different color.

4. Implementation

In the current phase of the research, we will only put our attention to build the foundation of the system: a web-based collaborative terrain editing. The system contains two main applications: client which is a terrain editor application, and server. To simplify the development process, our map editor is based on the work by Jasper Flick [24] with a few modifications.

4.1. Terrain Representation

Creating a natural terrain model is complex and requires various features. While the objective of the research is to create a usable terrain, in the current version we only focus on the development of client-server architecture instead of applying these features. We only applied the very basic terrain information where a terrain is defined as a two-dimensional array of integers. The indexes of the array define a point on a terrain and the integer value defines the height of the terrain at the corresponding point. The image below is the example of a terrain with size of 5x5. While the array indexing uses a standard 2-dimensional index, we need to modify the terrain indexing where the x index is shifted so hexagonal with the same x index fall in a straight line.

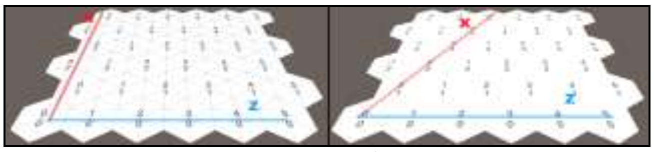

Fig. 2: Terrain Representation using set of hexagons with normal coordinates (left) and axial coordinates (right). Image courtesy of Flick [24].

The map editor allows user to modify the height of each hexagon by using a mouse click or touch input. Based on the position of the pointer, the system will modify the height to a certain value. To avoid hole between two hexagons with different height, the system creates a set of terraces to fill in the gap. The image below shows structure of terraces between two hexagons and the map with the application of terraces.

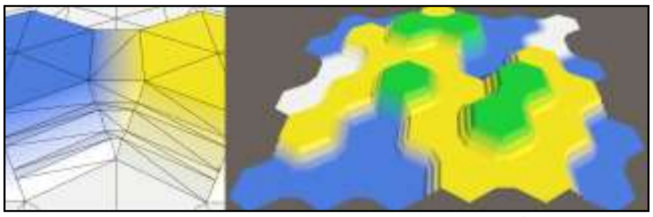

Fig. 3: Terraces between two polygons. Image courtesy of Flick [24].

While using some irregularities in building the terraces may increase the naturality of the terrain, it may cause differences amongst clients as we haven't yet to implement a single source random generator.

4.2. Client and Server Application

The system consists of two part of application: the client and the server. The client is used by the user to modify the map and the client will synchronize and distribute the changes.
User can change the height of the terrain at a certain point/hexagon. Each client will be assigned to a single color. In communicating with the server, we have to create a thread since Unity forbids network communication in the main thread. However, since Unity is not a safe-thread, it is also impossible for the communication thread to modify the terrain's cell since the cell grid is part of the main thread. To solve this problem, we created a pool of incoming message where the communication thread will store incoming messages from the server. These messages will be processed regularly by the main thread.

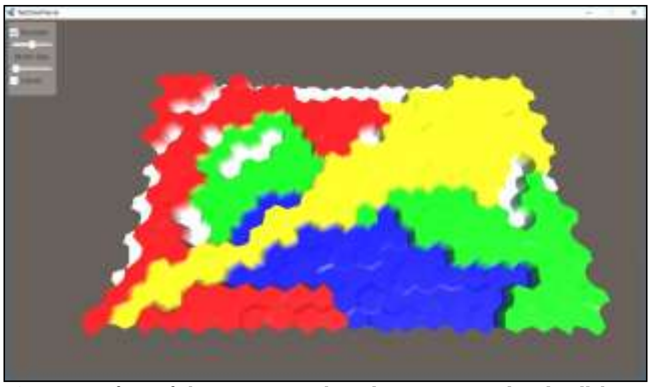

Fig. 4: Interface of the program with 4 clients connected and collaboratively editing in real time.

We built a basic server application with C++. The role of the server is only to listen actions from each client and distribute the actions to all clients. The server is capable to communicate with 10 clients concurrently. The server also has the capability to store 10000 previously distributed messages which is a crucial feature in massage synchronization.
As previously mentioned, the terrain data is organized as a two-dimensional integer array. This array is kept on the server storage. When a client is successfully connected to the server, this data will be transmitted to the client before the client is able to send a message.
As previously mentioned, the terrain data is organized as a two-dimensional integer array. This array is kept on the server storage. When a client is successfully connected to the server, this data will be transmitted to the client before the client is able to send a message.

4.3. Communication Protocol

The communication between client and server is using string-based message. When a client makes a change on the terrain, it will send the server the point of the terrain changed and the elevation. The format of the message is as follows:

EC 16 30 5

where EC is a code for elevation changed followed by the 2 value for terrain index and the height of the new elevation. The message will be processed and indexed by the server. If the message is valid, the server will distribute the indexed message to all clients with this format:

3 EC 16 30 5

Where the first integer value is the index of the message followed by the original message. The index is important to ensure the consistency of the content amongst all connected users. The protocol of this communication follows the diagram below.

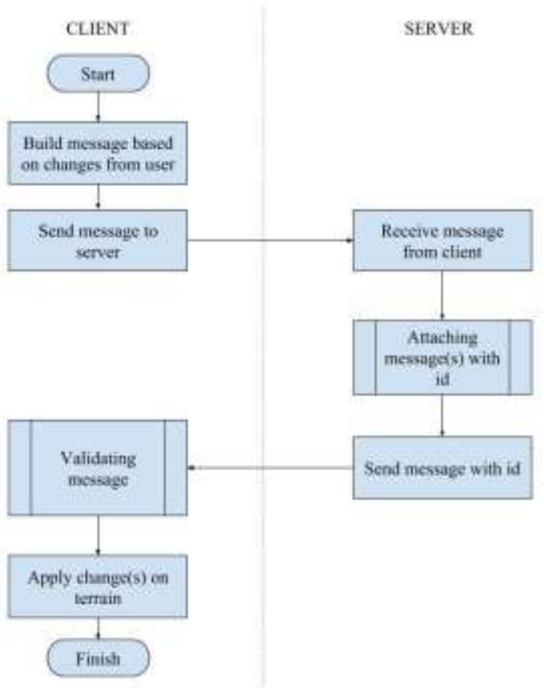
CLIENT
SERVER
Start
Build message based on changes from user
Send message to server
Receive message from client
Attaching message(s) with id
Send message with id
Validating message
Apply change(s) on terrain
Finish

Fig. 5: Communication Protocol between Client and Server.

Upon receiving the sorted actions, the client will have to verify while the changes are valid to be applied. Messages received by the client should be sorted based on the message id. If the incoming message does not contain the correct sequence, the client does not apply the change immediately. Instead, the client have to ensure that the last message is sequential with the incoming message. If the message is not valid, the client will send a request for the server to resend the previous message with a specific id. The request format is as follows:

RM 5

where RM is the code for resend message followed by the id of the missing message. If there are multiple messages missing on the sequence, the client simply sends multiple RM messages to the server. When the server receives this message, it will simply send the original message distributed to clients. By following this protocol, all clients are guaranteed to have a sequential Figure 5 shows the synchronization process. However, it is important to notice that there is a limit of messages kept in the server as previously stated.

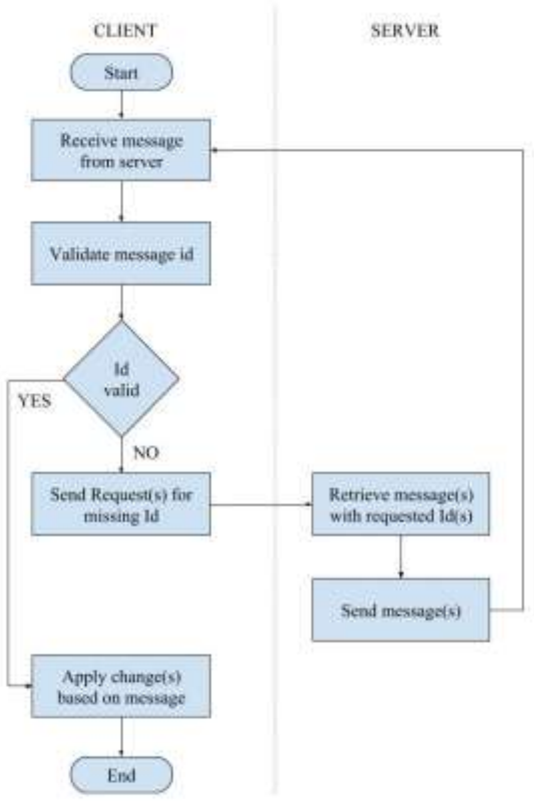
CLIENT
SERVER
Start
Receive message from server
Validate message id
Id valid
YES
NO
Send Request(s) for missing Id
Retrieve message(s) with requested Id(s)
Send message(s)
Apply change(s) based on message
End

Fig. 6: Message Synchronization between client-server.

4.4. Performance Test

To evaluate the performance of the system, we perform a test and observe the usability of the system. We used 4 computers acting as clients connected to a server. The connection between computers is through WiFi-based network. We focus our observation into two variables: graphic performance and network performance.

To measure the graphic performance, we collected and observed frame per second during runtime. We also extended the terrain to increase the polygon count. We use a device with gaming specification (Intel Core i7 Quad Core, 16 GB DDR4, GTX 1070 6GB GDDR5) for the graphic performance test. Based on our observation, using a terrain with the size of 20x15 (with 300 points on terrain) gives above 120 frame per second performance. Extending the size up to 80x60 (4800 points) still gives a solid performance at 63 frame per second averagely. We noticed that the system has significant performance drop at 160x120 (19.200 points) where the recorded performance was dropped to 42 frame per second.

The second variable we observed during the test is the input delay. Since all user changes has to be sent and sorted by the server, there will be a delay between user's input and the changes on the terrain. We noticed that the delay is minimum with less than 100 millisecond input delay. However, it is important to notice that the test only involved 4 clients. Hence, it is premature to claim that the network is optimal.

5. Conclusion

In this paper, we presented our work in building a client-server collaborative terrain editor. We have successfully tested the application by using 3 different devices acted as clients connected to a server. Based on the test, we found that the system is working properly with a good degree of usability. There was no significant delay problem found in the test. Moreover, the system has managed to keep a synchronized data amongst all client.

While we are satisfied with the networking feature of the system, there terrain representation is not yet usable for commercial use. Creating a natural terrain requires a lot of features and in the future project, we would like to add more work on the terrain. We would like to focus our work on adding irregularities on the terrain generation and add procedural method in generating the details.

Acknowledgement

This research is supported by Direktorat Riset dan Pengabdian Masyarakat (DRPM), Kementerian Riset, Teknologi, dan Pendidikan Tinggi, Indonesia and organized by Lembaga Penelitian, Universitas Sumatera Utara. We would like to thank for the support from Lembaga Penelitian USU. We would also like to thank Faculty of Computer Science and Information Technology and Faculty of Mathematics and Natural Science, Universitas Sumatera Utara for all the support during the process of this research.

References

[1] B. Beneš, “Physically-based hydraulic erosion,” 2006, pp. 17-22 PAGE@5.

[2] N. H. Anh, A. Sourin, and P. Aswani, “Physically based hydraulic erosion simulation on graphics processing unit,” 2007, p. 257.

[3] H. Zhang, D. Qu, Y. Hou, F. Gao, and F. Huang, “Synthetic Modeling Method for Large Scale Terrain Based on Hydrology,” *IEEE Access*, vol. 4, pp. 6238–6249, 2016.

[4] F. K. Musgrave, C. Kolb, and R. S. Mace, “The Synthesis and Rendering of Eroded Fractal Terrains,” *ACM SIGGRAPH Computer Graphics*, vol. 23, pp. 41–50, Feb. 1998.

[5] D. Fletcher, Y. Yue, and M. A. Kader, “Challenges and Perspectives of Procedural Modelling and Effects,” 2010, pp. 543–550.

[6] R. Smelik, T. Tutenel, K. J. de Kraker, and R. Bidarra, “Integrating procedural generation and manual editing of virtual worlds,” 2010, pp. 1–8.

[7] G. J. P. de Carpentier and R. Bidarra, “Interactive GPU-based procedural heightfield brushes,” 2009, p. 55.

[8] [8] A. Cristea and F. Liarokapis, “Fractal Nature - Generating Realistic Terrains for Games,” 2015, pp. 1–8.

[9] “Planetside Software – The home of Terragen – Photorealistic 3D environment design and rendering software.” .

[10] “GAIA | Procedural Worlds.” [Online]. Available: http://www.procedural-worlds.com/gaia/. [Accessed: 05-Oct-2018].

[11] J. Grudin, “Computer-supported cooperative work: history and focus,” *Computer*, vol. 27, no. 5, pp. 19–26, May 1994.

[12] W. Reinhard, J. Schweitzer, G. Volksen, and M. Weber, “CSCW tools: concepts and architectures,” *Computer*, vol. 27, no. 5, pp. 28–36, May 1994.

[13] T. D. Palmer and N. A. Fields, “Computer supported cooperative work,” *Computer*, vol. 27, no. 5, pp. 15–17, May 1994.

[14] P. Dourish and V. Bellotti, “Awareness and coordination in shared workspaces,” 1992, pp. 107–114.

[15] J. Grudin, “Why CSCW applications fail: problems in the design and evaluationof organizational interfaces,” 1988, pp. 85–93.

[16] L. Makris, I. Kamilatos, E. V. Kopsacheilis, and M. G. Strintzis, “Teleworks: a CSCW application for remote medical diagnosis support and teleconsultation,” *IEEE Transactions on Information Technology in Biomedicine*, vol. 2, no. 2, pp. 62–73, Jun. 1998.

[17] S. K. Chin, A. W. Yeo, and N. Musa, “Towards applying CSCW in improving orthography system development process,” 2013, pp. 55–59.

[18] A. Kaasinen and Y. I. Yoon, “Mobile advertising model in N-Screen environment for CSCW,” in *2012 7th International Conference on Computing and Convergence Technology (ICCCT)*, 2012, pp. 140–143.

[19] Y. Kakehi, M. Iida, T. Naemura, Y. Shirai, M. Matsushita, and T. Ohguro, “Lumisight Table: an interactive view-dependent tabletop display,” *IEEE Computer Graphics and Applications*, vol. 25, no. 1, pp. 48–53, Jan. 2005.

[20] Y. Kitamura, T. Konishi, S. Yamamoto, and F. Kishino, “Interactive stereoscopic display for three or more users,” 2001, pp. 231–240.

[21] R. Klauck, S. Lorenz, and C. Hentschel, “Collaborative work in VR Systems: A software-independent exchange of avatar data,” 2016, pp. 133–136.

[22] Y.-U. Ha, J.-H. Jin, and M.-J. Lee, “Lets3D: A Collaborative 3D Editing Tool Based On Cloud Storage,” *International Journal of Multimedia and Ubiquitous Engineering*, vol. 10, no. 9, pp. 189–198, Oct. 2015.

[23] Jos Timanta Tarigan, Rahmat Widia Sembiring, Maya Silvi Lydia, Opim Salim Sitompul, Mahyuddin K.M. Nasution, and Muhammad Zarlis, “Application Architecture for Collaborative Terrain Editing,” in *Proceedings of 2017 the 7th International Workshop on Computer Science and Engineering (WCSE 2017)*, Malaysia, 2017.

[24] J. Flick, “Unity Hex Map Tutorials,” *Catlike Coding*. [Online]. Available: https://catlikecoding.com/unity/tutorials/hex-map/. [Accessed: 06-Jul-2018].