The Application and Optimization of Unity Image Fusion Algorithm in HTC Vive Controllers Interaction *

Lei Chen
Department of Computer Science and Engineering
Southern University of Science and Technology
China
chenl3@mail.sustc.edu.cn

Zongwei Luo
Department of Computer Science and Engineering
Southern University of Science and Technology
China
luozw@sustc.edu.cn

ABSTRACT

There is [1] an increasing acceptance level of virtual reality nowadays, owing to their great participation mechanism and wide range of application , such as game, education, health care and military. In this paper, a practical algorithm is introduced and extended from 2D mode to 3D mode by HTC Vive controllers. Result shows that Unity image fusion algorithm has a great application in virtual reality environment. For less delay and dynamic memory consumption when running, two algorithm optimization methods are raised and innovative combination between them is under consideration with continuous evolution of Unity engine.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics • **Networks** → Network reliability

KEYWORDS

Unity Image Fusion, HTC Vive, Controller Interaction, Algorithm Optimization

SIMUTOOLS '17, September 11–13, 2017, Hong Kong, China

ACM ISBN 978-1-4503-6388-4/17/09…$15.00
https://doi.org/10.1145/3173519.3173526

1 INTRODUCTION

Virtual Reality leads a trend nowadays , with its application penetrating many fields in science and everyday life. Gamification education ,and industrial simulation are bright directions. They all can be realized in 2D mode but more vivid in 3D mode. There are many head mount displays in the market for immersed 3D mode experience and HTC Vive has a relatively better interactive functions. So far, HTC Vive has sold more than 140,000 sets[1]. Using its controllers , game users can behave as if this pair of controllers were their hands and trigger dazzling effects ,such as instant transfer, vibration, pickup, trigger ,emission etc. The game designers also create new algorithm to make people feel more real ,such as doodling on the canvas. However, the final rendering effects depend on quality of the algorithm and its combination with controllers. So algorithm optimization is an important and uninterruptible job during virtual reality expression.

2 RELATED ALGORITHM APPLICATION IN VIRTUAL REALITY

Contents making and rendering is a crucial part that attracts lots of users to make relationship with virtual reality. For example, when to develop a VR game ,behavior tree is proper algorithm for managing a character in the game scene by defining main behavior classification for the character[2].Once the condition is reached, and then the characters can start corresponding action as if they have the intelligence to make a choice. Realtime panoramic video stitching can be seen as a branch of virtual reality which is based on image stitching algorithm. The steps are always like this: image projection transformation, feature extract, feature filtering , feature registration, and image fusion[3]. Some algorithms are used here, such as SIFT, RANSAC, KD-Tree, Laplacian Pyramid, ROD etc[4]. View bypass and time wrapping are often combined together when in rendering. View bypass[5] calculates the visual difference between the two inputs for modification and time wrapping[6] processes the visual matrix in rendering layer by capturing new data from the users and creating updated visual matrix instead of directly delivering the images from game layer to display layer. This operation minimizes the image rendering delay caused by users' action. Besides, some plugins in unity are powerful because of the functions covered in

them. For example, VRTK – SteamVR Unity Toolkit can be used in HTC Vive controller that realize visual and touch effects[7].

3 UNITY IMAGE FUSION ALGORITHM

The principle of Unity image fusion algorithm is to merge the two images, putting the pixels of one image on another[8]. Content of this project is based on the algorithm. First define a generic queue, obtain the texture coordinates of the base graph and then store them in the generic queue. Traverse region of the top figure with nested loops, obtain every pixel in the region that ought to appear and corresponds to the top figure , then get the color of the corresponding pixel point within the two figures, and finally fuse them. The algorithm is often used in the bullet trail making in shooting games. The relevant variables and objects are defined first, especially a generic queue *Queue<Vector2> uvQueues*[8] to store pixel information, and then obtain initialization value of variables, and next conduct bullet collision detection, bullet image traversal and pixel fusion. Related code is as follows:

Define a generic queue to store the information of image pixels.

Initialize: Initialize a bivector queue uv

Judge whether bullets hit wall, obtain hitpoint coordinates in Wall and enqueue

```
if Raycast hit outwards in Physics
    if collider's name == "Wall"
        give the value of text coordinates to uv
        enqueue uv
    end
end
```

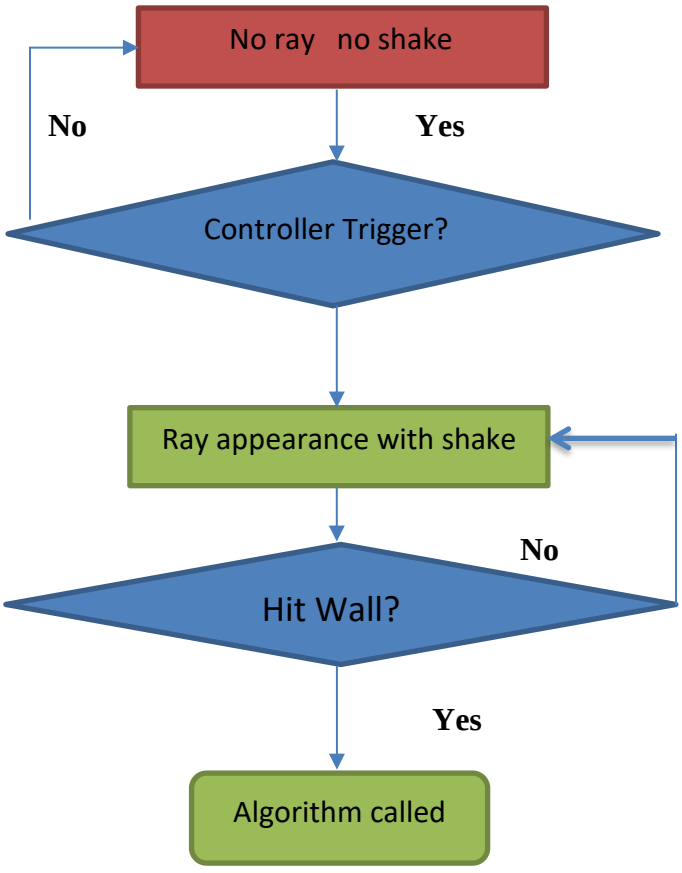
Traverse bullet image region, obtain pixels corresponding to every bullet , obtian color of every pixel in wall, obtain color of pixels corresponding to every bullet and finally complete fusion.

```
for each value within bullet width do
    for each value within bullet height do
        substract bullet width from wall width in uv plus i get w
        substract bullet width from wall height in uv plus j get h
        wall's color=get pixel of w and h from texture of new wall
        bullet's color = get pixel of i and j from texture of bullet
        set pixel to w and h with fusion of color of wall and bullet
    end
end
```

Before run the code, make some new setting on the images, such as texture type , read or write enabled, image maxsize and format (bullet maxsize 64 and wall 2048, their format all ARGB 32 bit)[8]. When in operation , attach script with this algorithm to GameObject Wall, click the mouse, and then the camera emits a ray which bring back the collision information, namely the *uv* coordination ranging from 0 to 1.

4 APPLICATION OF UNITY IMAGE FUSION ALGORITHM IN HTC VIVE CONTROLLERS INTERACTION

The mouse can trigger ray to create fused pixel, but it is only in a 2D mode which won't bring a sense of immersion (refer with: Fig. 1). So combination with HTC Vive is needed to boost the advance of experience. The algorithm described above within script *generateHole* is encapsulated in a function called *hitChange(Vector2 uv)* with the parameter providing texture coordinates of wall image. Make a function call here to run the fusion algorithm in the controller interaction script *HTCStick* when condition is satisfied. In controller interaction script, controllers and *raycasthit* information is defined as variables, as well as script *generateHole (gH)*. *LineRenderer* [9] is set to make ray visualized. A condition judgement is established here to complete corresponding logistics. For example, when the trigger is pressed down , function *TriggerHapticPulse (1200)* [7] is activated and controller starts shaking. *LineRenderer* is enabled at the same time. After setting *hitpoint* position, the ray begins to do its collision detection. Unity image fusion algorithm may be called once the condition is enough(refer with: Fig. 2,Fig.3). In another condition, namely trigger up , *LineRenderer* is disabled as if the ray is deleted. Related flow chart is as follows:



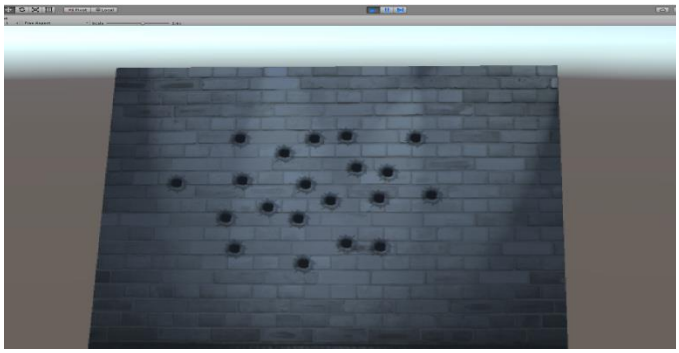


Figure 1: 2D image fusion

Figure 2. Figure 3: 3D image fusion (in VR environment)

5 ALGORITHM OPMIZATION

In this project ,script *generateHole* regarded as a component is added to GameObject *Wall*, so calling function *hitChange (uv)* must find the GameObject it belongs to firstly. Though in script *HTCStick* , *generatehole* is defined as a public variable, it can't be dragged to the corresponding variable position of *HTCStick* in inspector block. Method *GameObject go*[10] may be used to instantiate script *generatehole* as a new GameObject, so that it can be put into script *HTCStick* directly without going through prefab *Wall*, namely reducing dynamic memory establishing consumption. This is still a thought that is waiting for verification on condition that there may not be the syntax that functions in one script can be called by another script through instantiating the script as a prefeb in Unity[11]. However ,this is an innovative point that should be developed .

What's more, editing shader can realize less delay in texture rendering [12]. In this project , texture color is obtained by function *GetPixel()* [8]. Each time in a loop you take turn to get corresponding pixels of the two textures, save them in another variables repectively and then *SetPixel ()* [8] between the variables to realize pixel fusion .This is inefficient. If switching the texture very frequently during iteration, rendering delay can be obvious. if edit and save shader ,you can change the way of obtaining the related pixels in one line every time to getting textures's RGB every other line, then set custom/GH as shader type of needed material and finally add the two textures. Fusion efficiency can be changed a lot. Though I haven't apply shader editing here ,it is still a meaningful trend in the following optimization process of algorithm.

6 CONCLUSIONS

Unity is a powerful engine for virtual reality and game development, though it has some limitation. To realize more vivid effects, the combination of core algorithm and other elements is very important. In this project , if the two optimization methods mentioned above can be realized, the coordination and cooperation in traditional Unity image fusion algorithm and the two elements may create different experience. For example, edit the shader of GameObject Wall ,and then add related textures or use traditional algorithm but call hitChange() in a direct way[13] .If the architecture of Unity engine evolves in the future, may a brand new thought appear which ought to bring more high quality virtual reality experience.

A HEADINGS IN APPENDICES

A.1 Introduction

A.2 Related Algorithm Application in Virtual Reality

A.3 Unity Image Fusion Algorithm

A.4 Application of Unity Image Fusion Algorithm in HTC Vive Controllers Interaction

A.5 Algorithm optimization

A.6 Conclusions

A.7 References

ACKNOWLEDGMENTS

REFERENCES

[1] Available at http://www.sohu.com/a/111103460_473283.
[2] Available at http://www.52vr.com/article-884-1.html.
[3] Wai-Kwan Tang, Tien-Tsin Wong, and Pheng-AnnHeng, A system for real-time panorama generation and display in tele-immersive applications[J], IEEE Transactions on Multimedia, Volume 7, Issue 2, Apr 2005: 280 – 292.
[4] Szeliski, R.: Image alignment and stitching: A tutorial. Technical Report MSR-TR-2004-92, Microsoft Research (October 2004).
[5] Available at http://blog.sina.com.cn/s/blog_7e5c6673010117kt.html.
[6] Available at https://www.ithome.com/html/next/216624.htm.
[7] Available at http://blog.csdn.net/kitok/article/details/52836088.
[8] Available at http://www.jianshu.com/p/65fcae9687f0.
[9] Available at http://blog.csdn.net/zuoyamin/article/details/8997729.
[10] Available at http://tieba.baidu.com/p/3105551323.
[11] Available at http://blog.csdn.net/smilingeyes/article/details/17767269.
[12] Available at http://tieba.baidu.com/p/3172432200.
[13] Unity Technologies, Unity 5.x from entry to mastery, Beijing, 2016.