Journal of The Korea Society of Computer and Information
Vol. 24 No. 4, pp. 51-56, April 2019

www.ksci.re.kr
https://doi.org/10.9708/jksci.2019.24.04.051

Real-Time Generation of City Map for Games in Unity with View-dependent Refinement and Pattern Synthesis Algorithm

Jong-Hyun Kim*

Abstract

In this paper, we propose an algorithm that can quickly generate and synthesize city maps in racing games. Racing games are characterized by moving a wide map rather than a fixed map, but designing and developing a wide map requires a lot of production time. This problem can be mitigated by creating a large map in the preprocessing step, but a fixed map makes the game tedious. It is also inefficient to process all the various maps in the preprocessing step. In order to solve this problem, we propose a technique to create a terrain pattern in the preprocessing process, to generate a map in real time, and to synthesize various maps randomly. In addition, we reduced unnecessary rendering computations by integrating view-dependent techniques into the proposed framework. This study was developed in Unity3D and can be used for various contents as well as racing game.

▸Keyword: Generation of city map, Pattern synthesis, View-dependent refinement, Unity3D

I. Introduction

최근에 인터랙션의 중요성이 커짐에 따라 사용자의 움직임을 고려한 콘텐츠 개발이 증강/가상현실, 모바일 앱, 게임 등에서 활발하게 이루어지고 있다[1,2](Fig.1 참조).

Fig. 1. Terrain and city map used in games[4].

대부분이 게임에서 이동의 자유도를 높이고자 넓은 지형을 원하지만 고정적인 맵으로 인해 콘텐츠를 지루하게 만든다. 전처리 과정에서 일시적으로 맵을 다양하게 만들 수 있지만 리소스나 계산시간 측면에서 비효율적이며, 이 문제를 해결하기 위해 좀 더 쉽게 지형을 생성하는 기법들이 게임엔진 내 에셋을 통해 개발되었지만, 큰 지형을 제작하기에는 충분하지 않다.

일반적으로 지형이나 도시 맵을 만들기 위해서는 2차원 높이 맵을 기반으로 사용자가 지형의 높낮이를 조절한다. 이 같은 방법은 직관적으로 지형을 모델링 할 수 있다는 장점이 있지만, 제한된 영역 내에서 지형 편집의 용이성을 개선시켰다는 점 일뿐, 넓은 지형을 제작하려면 많은 시간과 노동이 필요하게 된다. 이 같은 문제는 도시 맵을 생성할 때 더욱더 어려워진다. 실제 지형의 높낮이 뿐만 아니라 도로 및 건물 배치까지 고려해야 하기 때문이다. 지형이나 텍스처를 만들기 위한 다양한 합성 기법들이 존재하지만 게임에 적용하기에는 계산비용이 매우 클 뿐 아니라[3], 이 접근법은 도로와 지형 내 존재하는 빌딩까지 합성하기에는 충분하지 않다.

제안하는 방법은 자동차가 넓은 지형을 움직일 수 있도록 자

• First Author: Jong-Hyun Kim, Corresponding Author: Jong-Hyun Kim
*Jong-Hyun Kim (jonghyhunkim@kangnam.ac.kr), Dept. of Software Application, Kangnam University
• Received: 2019. 02. 18, Revised: 2019. 03. 27, Accepted: 2019. 04. 04.
• This paper is an extension of the paper presented("Terrain pattern synthesis method for quick city map generation in racing game") at the conference(Proceedings of the Korea Society of Computer and Information Conference 2018) (Best Paper Award).

동으로 교차로와 도로뿐만 아니라 주변 빌딩까지 끊임없이 생성/합성하여 고해상도 도시 맵을 생성한다. 이를 수행하기 위한 본 연구의 장점은 아래와 같다.

▪ 큰 도시 맵을 실시간으로 생성: 게임 오브젝트(또는, 사용자)의 위치를 기반으로 로컬하게 지형을 합성함으로써 빠르게 도시 맵을 생성할 수 있다.

▪ 뷰 의존형 기반의 지형 합성: 지형과 빌딩을 합성하는 과정에서 게임 오브젝트가 바라보거나 움직이는 방향만 합성할 수 도록 최적화하여 성능을 개선시킨다.

II. Preliminaries

1. Related works

이미지를 이용한 예제 기반 텍스처 합성 연구에는 다수의 알고리즘들이 다양한 접근 방법을 통해 개발되어 왔다. 대부분의 연구들은 하나의 예제를 사용하여 적당히 크기의 동질적 텍스처 합성 결과물을 생성하는데 그 목적을 두고 있다[3,5,6,7]. 몇몇 이미지 합성 알고리즘들이 비동질 예제를 이용하여 임의의 텍스처를 합성하거나 여러 개의 예제 입력으로부터 하나의 텍스처를 생성해 내는 시도를 하였다[6,8,9]. 하지만 이런 이미지 기반 접근법들은 3차원 지형 합성이나 건물 합성에는 적합하지 않은 구조로 설계되어 있는 경우가 대부분이다.

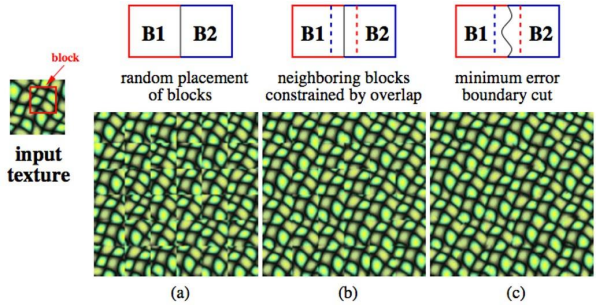


Fig. 2. Texture synthesis[3].

Lefebvre와 Hoppe는[3] 특징 기반의 드래그-앤-드롭 방식을 이용하여 예제 이미지의 임의의 특징점을 사용자가 원하는 대로 배치할 수 있도록 하고, 공간 결정적 합성 기법을 활용한 대형 텍스처 생성을 소개하였지만(Fig. 2 참조), 한 장의 예제 이미지만을 사용하여 동질적인 합성 결과물만을 생성하는 제한점이 있다. 한 장의 예제는 동질적인 3차원 지형 맵을 생성하기에는 충분하지만 좀 더 복잡한 재질을 갖는 지형이나, 다양한 건물 패턴과 도로를 갖는 맵을 생성하기에는 부족한 방법이다. Rosenberger 등은[10] 비동질적 텍스처를 입력 받아 이를 색상을 기반으로 한 다수 개의 계층적 층으로 분류하여, 원본 예제의 분포를 크게 해치지 않는 범위 내에서 비동질적 텍스처 합성 결과물을 생성하였다. 하지만 공간결정적 합성기법의 부재로 인하여 대형 텍스처 생성에 적합하지 않은 구조를 가지고 있으며, 이 같은 문제는 큰 지형/도시 맵을 생성하는 프레임워크에서도 동일하게 나타난다.

Hertzmann 등은[8] 외부 이미지 입력을 통해 슈퍼 해상도급 합성 결과물을 만들 수 있지만 합성 결과물의 크기에 비례하여 하드웨어 자원을 필요로 하는 문제가 있다. 이 같은 문제는 3차원 지형으로 넘어갈수록 더욱더 두드러지게 나타난다. Han 등은[11] 여러 개의 텍스처를 활용하여 계층구조를 가진 비동질적 대형 텍스처를 생성하는 기법을 제안했다. 하지만, 이 연구에서 사용한 방법은 전체 구조 가이드를 위한 텍스처를 한 장 이상 반드시 입력해야 하고, 하위 텍스처군 간의 접점이 없는 경우에는 제대로 된 결과물을 얻을 수 없다. Park 등은[12] 전환 이미지 생성, 인덱스 점핑 단계를 도입하고 다수 장의 예제로부터 비동질적 특성을 지니는 다중 텍스처 합성 결과물을 만들었다. 하지만 대형 텍스처 합성 결과물을 만드는데 전체적인 텍스처 요소의 분포를 고려하지 않아서 인위적인 느낌의 결과물이 생성되는 문제가 있다.

Fig. 3. Adaptive synthesis of distance fields[13].

Lage 등은[13] 텍스처 합성 기법을 활용하여 거리장을 기반으로 3차원 모델을 합성하였고, Lee 등은[14] 지오메트리 합성을 적응형 프레임워크로 확장함으로써 3차원 모델의 합성 품질 및 계산 성능을 개선시켰다. 하지만, 이런 합성 접근법들은 랜덤 기반으로 생성되기 때문에 결과 품질에 노이즈가 나타난다. 뿐만 아니라 사용자의 의도에 맞는 모델을 생성하는 것이 어렵고, 계산시간이 오래 걸리기 때문에 게임과 같은 실시간 애플리케이션에 적용하기에는 적합하지 않다.

최근에는 전통적인 이미지 합성 기법을 사용하지 않고 인공지능 기법인 Variational auto-encoder(VAE)[15]와 Generative adversarial network(GAN)[16]을 활용하여 직접적으로 이미지를 합성하는 연구들이 제안되었다. 일반적인 Vanilla GAN 기법은[16] 노이즈 없는 이미지 합성기술을 제안했지만, 사용자가 쉽게 제어할 수 없는 구조이다. 이 한계점을 해결하고자 제안된 Condition GANs은 단순한 노이즈 이외에 사용자 입력 방식을 기반으로 이미지를 합성하기 때문에 결과물을 쉽게 제어할 수 있다[17].

II. The Proposed Scheme

본 논문에서 제안하는 방법은 게임 오브젝트의 위치나 속도를 고려하여 적응적으로 지형과 빌딩을 포함한 도시 맵을 실시간으로 합성할 수 있는 새로운 프레임워크를 제안한다. 본 연구의 방법은 아래와 같은 순서로 수행된다.

1) 시작 시 게임 오브젝트의 위치를 중심으로 작은 크기의 도로 및 빌딩이 생성됨
2) 게임 오브젝트가 움직이면 그 방향으로 도로와 빌딩이 실시간으로 생성됨
3) 게임 오브젝트가 정해진 방향이 아닌, 자유롭게 움직여도 모든 도로 방향과 건물들이 막히지 않고 연결될 수 있도록 합성됨
4) 게임 오브젝트가 이동하면서 바라보는 시점만 시각화하여 수행속도를 최적화하고, 결과적으로 보이지 않는 부분은 비활성화시켜 큰 도시 맵을 생성하는데 소요되는 계산 시간을 줄인다.

1. Preprocessing

제안하는 방법은 도시 맵 합성을 효율적으로 계산하기 위해 전처리 과정에서 도로, 교차로, 빌딩 등 4가지 종류의 프리팹(Prefabs) 생성한다(Fig. 4 참조). 큰 지형의 도시 맵을 제작하기 위해서는 많은 수의 패턴과 오브젝트가 필요한데 이 과정에서 본 논문에서는 오브젝트를 런타임에 인스턴스화하여 오브젝트의 재사용성이 높이려고 Unity의 프리팹을 이용하였다.

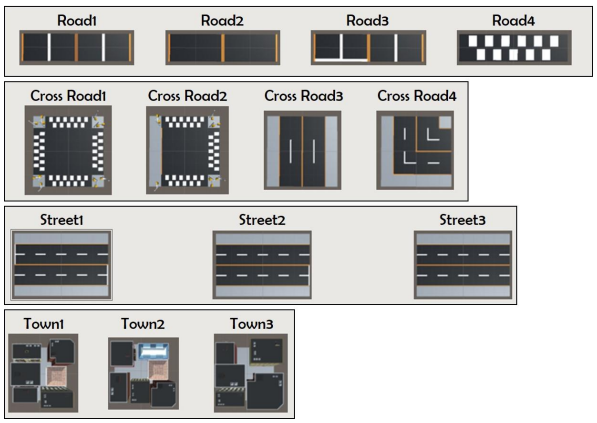


Fig. 4. Various types of prefabs.

2. Road Generation

본 연구에서는 도로를 랜덤하게 생성하기 위해 2가지 배열을 사용한다. 첫 번째는 같은 자리에 또 다시 오브젝트를 생성하는 일이 없도록 제한하기 위한 오브젝트 배열이다. 두 번째 배열은 생성된 오브젝트의 진행 방향을 저장할 배열이다. 도로를 생성할 때는 3개의 상태 변수(벽, 횡당 보도, 직진 도로)를 사용하였으며, 우리는 교차로를 먼저 생성한 후 교차로 사이를 도로와 건물로 이어준다.

Fig. 5는 격자 기반으로 건물도 도로를 연결해주는 알고리즘이다. 본 연구에서는 5x5 격자의 크기를 이용하였으며, 초기에는 Fig 5a와 같은 형태로 건물 모델인 Town을 한칸 건너 한칸씩 생성시킨다. 이 같은 구조는 Fig. 5a에서도 쉽게 볼 수 있으며, 빨간색 화살표처럼 대각선 네 방향에 Town 모델이 존재한다면 교차로 모델인 Cross road를 배치시킨다. 여기까지 진행되면 Fig. 5b와 같은 형태로 나타나며, 나머지 빈칸들은 Street 모델로 채워준다 (Fig. 5c 참조). 결과적으로 4개의 건물들 사이에 있는 공간에는 교차로 모델인 Cross Road를 이어주고, 교차로들 사이에는 직진도로 모델인 Street로 연결시킨다.

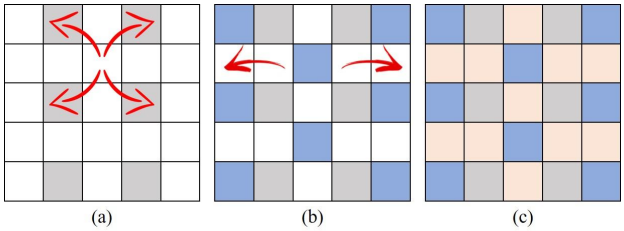


Fig. 5. Grid based city map generation (gray : town model, blue : cross road model, pink : street model).

최종적인 결과는 Fig 5c와 같은 배열구조가 나타나게 되며, 5x5 격자구조를 갖게 된다 (Fig. 6a 참조). 도시 맵을 초기에 생성했고, Fig. 6b에서 보듯이 온라인 과정에서 자동차가 옆으로 이동한다면 자동차 뒷부분에 해당하는 영역은([4,0]에서 [4,4]까지) 필요 없기 때문에 격자공간에서 삭제해주고 (Fig. 6b에서 검정색 화살표 참조), 진행방향으로 건물과 도로를 계속적으로 합성한다 (Fig. 6c 참조).

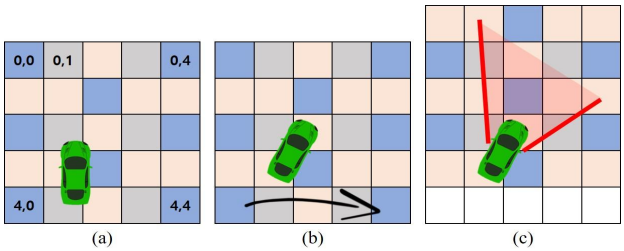
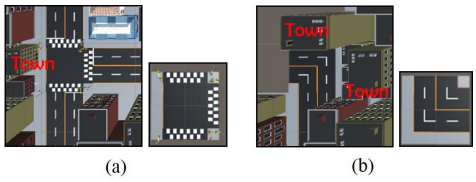


Fig. 6. City maps updated by movement of car (arrow : delete areas, red area : view-dependent area).

격자 공간에서 모델을 추가시킬 때 Town, Street 모델같은 경우에는 랜덤하게 3가지 종류 중에 하나를 선택해서 생성해주지만, 교차로 모델인 Cross road는 몇 가지 예외처리가 필요하다. 건물을 합성하는 과정에서 이전에 건물이 이미 생성된 경우, 건물을 뚫고 지나갈 수 없기 때문에 Turn을 하는 형태로 교차로를 합성해야 한다.



(a) (b)

Fig. 7. Blocked cases.

Fig. 7은 주변 건물에 의해 막혀진 상황을 보여주는 결과이다. Fig. 7a에서 보듯이 왼쪽이 건물이 막혀진 경우 Cross Road2 모델을 사용하여 교차로를 넣어주고, 만약에 오른쪽 방향이 건물로 막혀진 경우에는 Cross Road2를 회전시켜 적용하는 방식을 이용한다. Fig. 7b는 건물로 막혀진 방향이 2개인 경우이며, 본 논문에서는 Cross Road3 모델을 이용하여 Turn 형태로 도로를 합성하였다. 결과적으로 막힘에 의한 상황까지 고려가 되면 모든 상황에 대해서 도로를 합성할 수 있게 된다.

제안하는 방법은 격자 기반으로 합성한 도시 맵을 생성하고, 이를 좀 더 최적화하기 위해 사용자 시점 기반 렌더링을 이용한다. Unity에서 제공하는 View-dependent 렌더링 기법과 통합하기 위해 우리는 메인 카메라가 바라보는 방향에 포함되어 있지 않은 오브젝트 렌더링을 끄는 방법을 사용한다. 이 기법은 Fig. 6c에서 보듯이 자동차가 바라보는 방향으로 렌더링을 하기 때문에 좀 더 빠른 결과를 얻을 수 있으며, 게임 내 FPS를 개선시킨다.

Fig. 8은 제안하는 방법을 이용하여 도시 맵을 구성한 결과이다. 앞에서 설명했듯이 격자 기반으로 도시 맵을 생성하기 때문에 Scene view에서 격자의 형태가 남아 있다 (Fig. 8에서 빨간색 참조). 하지만 지역적으로 게임 오브젝트 근처에서 도시 맵을 합성하기 때문에 사용자 시점인 Game view에서는 격자의 형태나 전체 맵의 형태가 보이지 않고 자연스럽게 게임을 즐길 수 있다.

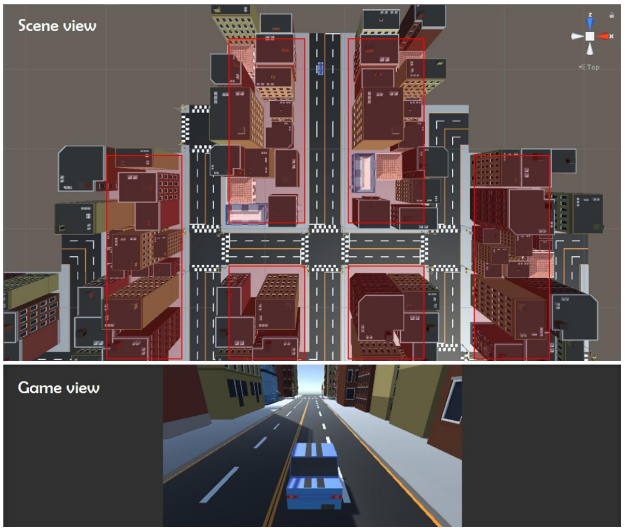


Fig. 8. Init scene with random generation of road and town.

IV. Results and Conclusions

본 연구에서는 Unity를 이용하여 개발하였으며 자동차의 위치를 기준으로 도시 맵을 지역적으로 생성/합성하는 결과를 만들어 냈다. 결과의 우수성을 보여주기 위해 우리는 전처리 과정에서 맵을 만들어서 사용하는 기법과 결과를 비교하였다.

우리는 전처리 과정에서 40×40 격자 크기를 갖는 큰 도시 맵을 만들어 놓고 레이싱 게임을 시작했으며 (Fig. 9 참조), 렌더링 뷰에서 보듯이 도로와 빌딩이 자연스럽게 연결되었지만 큰 도시 맵을 매 프레임 처리해야 되기 때문에 게임 실행이 느려 실시간 처리가 힘들다. 자동차의 움직임은 키보드를 통해 사용자가 입력하여 조절하였다.

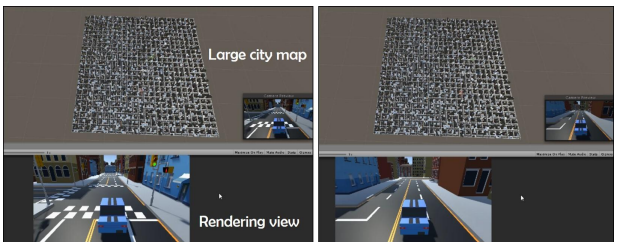


Fig. 9. Large city map calculated during preprocessing.

Fig. 10은 자동차 움직임에 따라 5×5 격자 크기의 도로를 랜덤하게 생성하고 경계부분을 또 다른 연결 도로 자연스럽게 합성한 결과이다. Fig. 9와는 다르게 지역적으로 도시 맵을 생성했음에도 불구하고 끊임없이 연결된 도로 맵을 만들었으며, Fig. 10에 비해 게임 속도가 10배 이상 빨라졌다.

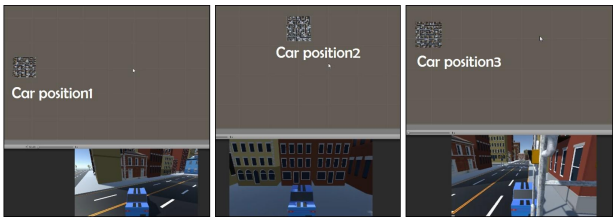


Fig. 10. City map generation with our method.

Fig. 12는 Fig. 10과 동일한 환경에서 View-dependent 렌더링 기술을 추가로 통합한 결과이다. 카메라의 Field of view를 45도로 설정하여 View-dependent에 이용하였으며 Fig. 11에서 보듯이 View 범위에 들어온 객체들만을 렌더링하여 효율성을 개선시켰다. 동일한 결과를 제작하는데 있어 Fig. 10보다 게임 속도가 1.5배 정도 빨라졌다.

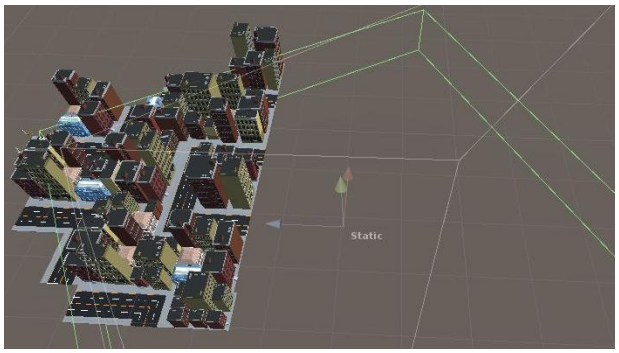


Fig. 11. View-dependent area with virtual camera.

본 연구에서는 Unity 게임 엔진 상에서 도시 맵을 합성하는 새로운 프레임워크를 제안했다. 높이 맵을 기반으로 지형을 생성하는 기존 접근법은 Fig. 9에서 제시한 결과와 비슷하며, 결과적으로, 높이 맵을 생성하는 기반 폴리곤 메쉬의 크기와 해상도에 의존하는 결과를 만들어 낸다. 하지만, 우리는 문법기반으로 합성하기 때문에 기반 폴리곤 메쉬에 독립적으로 수행되며 지형뿐만 아니라 도로, 건물 등 좀 더 높은 자유도를 갖는 합성 결과를 보여주었다. 향후 커브 형태나, 지하도처럼 좀 더 복잡한 도로를 합성 할 수 있는 기법에 대해 연구하여 게임, 교육, 가상현실 콘텐츠 등 다양한 분야에 적용해볼 계획이다.

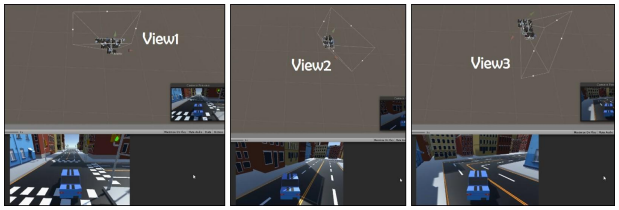


Fig. 12. City map generation with our method and view-dependent method.

REFERENCES

[1] Cakmak, Tuncay and Hager, Holger, “Cyberith Virtualizer: A Locomotion Device for Virtual Reality”, ACM SIGGRAPH, pp.6:1-6:1, 2014.

[2] Sutherland, Ivan E., “A Head-mounted Three Dimensional Display”, Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I, pp.757-764, 1968.

[3] Lefebvre, Sylvain and Hoppe, Hugues, “Parallel Controllable Texture Synthesis”, ACM SIGGRAPH, pp.777-786, 2005.

[4] Gameloft, “Asphalt Xtreme” video game, https://asphaltxtreme.com/, October 2016.

[5] Lefebvre, Sylvain, and Hugues Hoppe. "Appearance-space texture synthesis." ACM Transactions on Graphics (TOG). Vol. 25. No. 3. ACM, 2006.

[6] Bar-Joseph, Ziv, et al. "Texture mixing and texture movie synthesis using statistical learning." IEEE Transactions on Visualization & Computer Graphics, pp.120-135, 2001.

[7] Efros, Alexei A., and William T. Freeman. "Image quilting for texture synthesis and transfer." Proceedings of the 28th annual conference on Computer graphics and interactive techniques. ACM, 2001.

[8] Hertzmann, Aaron, et al. "Image analogies." Proceedings of the 28th annual conference on Computer graphics and interactive techniques. ACM, 2001. ACM, 2001.

[9] Heeger, David J., and James R. Bergen. "Pyramid-based texture analysis/synthesis." Proceedings of the 22nd annual conference on Computer graphics and interactive techniques. ACM, 1995.

[10] Rosenberger, Amir, Daniel Cohen-Or, and Dani Lischinski. "Layered shape synthesis: automatic generation of control maps for non-stationary textures." ACM Transactions on Graphics (TOG) 28.5 (2009): 107.

[11] Han, Charles, et al. "Multiscale texture synthesis." ACM Transactions on Graphics (TOG). Vol. 27. No. 3. ACM, 2008.

[12] HanWook Park, HaeWon Byun and ChangHun Kim, 2011, "Transition Image Generation for Multi-input Texture Synthesis," Journal of KIISE : Computer Systems and Theory, Vol. 38, No. 4, pp. 202~206, 2011

[13] Lagae, Ares, Olivier Dumont, and Philip Dutre. "Geometry synthesis by example." Shape Modeling and Applications, 2005 International Conference. IEEE, 2005.

[14] Lee, Sung-Ho, et al. "Adaptive synthesis of distance fields." IEEE transactions on Visualization and Computer Graphics Vol. 18.7, pp. 1135-1145, 2012.

[15] Lalonde, Jean-François, Derek Hoiem, Alexei A. Efros, Carsten Rother, John Winn, and Antonio Criminisi. "Photo clip art." In ACM Transactions on Graphics (TOG), vol. 26, no. 3, p. 3. ACM, 2007.

[16] Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." In Advances in neural information processing systems, pp. 2672-2680. 2014.

[17] Mirza, Mehdi, and Simon Osindero. "Conditional generative adversarial nets." arXiv preprint arXiv: 1411.1784, 2014.

Authors

Jong-Hyun Kim received the B.A. degree in the department of digital contents at Sejong University in 2008. He received M.S. and Ph.D. degrees in the department of computer science and engineering at Korea University, in 2010 and 2016.Prof. Kim is an assistant professor in the department of software application in Kangnam University. His current research interests include fluid animation and virtual reality.