Journal of KIIT. Vol. 19, No. 1, pp. 139-146, Jan. 31, 2021. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2021.19.1.139

로그 라이크 게임에서 새로운 맵 생성 알고리즘

김재현*, 이부형**

New Map Generation Algorithm in Rogue-Like Game

Jae-Hyun Kim*, Boo-Hyung Lee**

요 약

로그 라이크(Rogue-Like)게임에서 맵 복잡도 및 랜덤성의 증가는 게이머의 흥미를 유발시키는 중요한 요인이다. 본 논문에서는 절차적생성기법(PCG)을 이용하여 로그 라이크 게임용 맵을 생성하지만 맵 내 다양한 구조의 템플릿을 추가하여 맵의 랜덤성 및 복잡도를 증가시킬 수 있는 새로운 맵 생성 기법을 제안한다. 제안한 방법은 템플릿을 이용한 트리구조 확장 방식이며, 맵의 복잡도는 맵의 시작위치와 최종위치간의 최단거리를 구한 후 최단거리 내에 포함된 노드의 개수 및 방향전환회수를 이용하여 성능을 평가하였다. 노드의 개수에서 제안된 방법인 템플릿이용 트리구조 확장 방식이 BSP방식에 비해 평균 17.5%, 트리구조 확장 방식에 비해 9% 증가되었으며, 방향전환횟수에서는 BSP에 비해 평균 85%, 트리구조 확장 방식에 비해 20%이상 증가됨을 보였다. 또한 매회 템플릿이 변화됨과 동시에 새로운 맵이 만들어짐으로써 자연적으로 랜덤성 효과가 발생됨으로 보였다. 제안된 알고리즘은 Unity, VisualStudio, GIT의 GUI툴인 Git Kraken을 이용하여 구현하였다.

Abstract

Map complexity and randomness in Rogue-like Game are very important factor which can arouse an interest of gammer. In this paper, we propose the new map generation technique which is can enhance Map complexity and randomness by adding templates having various structures in the map generated by known procedural Content Generation(PCG) method. The proposed method can be mentioned the extended tree structure using templates and map complexity the proposed method was evaluated by the number of nodes in the shortest path and number of direction turuns between starting position and exit position in the map. The number of nodes of the proposed method was increased 17.5% and 9% each in comparison to the BSP and extended tree structure method. Also, the number of direction turns was enhanced 85% and 20% each in comparison to BSP and extended tree structure method. The method was implemented using Unity, VisualStudio, Git Kraken(GUI tool of GIT)

Keywords

rogue-like game, map complesxity, map randomness, gammer, extended tree structure using templates

* 공주대학교 컴퓨터공학부
- ORCID: https://orcid.org/0000-0003-1009-3928
** 공주대학교 컴퓨터공학부 교수 (교신저자)
- ORCID: https://orcid.org/0000-0003-4434-8933

· Received: Dec. 07, 2020, Revised: Jan. 20, 2021, Accepted: Jan. 23, 2021

· Corresponding Author: Boo-Hyung Lee
Dept. of Computer Engineering & Science, Kongju National University, Budae-dong, Cheonan-si, Chungcheongnam-do, 330-717, Korea,
Tel.: +82-41-521-9232, Email: BHL1998@kongju.ac.kr

Ⅰ. 서 론

로그 라이크(Rogue-Like)는 MS-DOS 시절에 만들어진 Rogue라는 게임에서 영감을 받아 로그와 같다(Rogue + Like)라는 의미에서 붙은 게임 장르의 이름이다[1]. 이 게임의 특징은 첫째, 목숨은 언제나 “하나의 목숨”만으로 게임을 진행하게 된다는 점이다. 게임을 진행했던 시간과 관계없이 게임이 끝나는 순간 처음부터 게임을 다시 시작해야하는 것을 의미한다. 둘째, 랜덤성이다. 즉, 매 게임마다 랜덤하게 만들어진 맵에서 게임을 진행하게 되고 획득할 수 있는 아이템 또한 랜덤하다는 특징을 갖는다. 따라서 로그 라이크 게임은 한번만 죽어도 게임을 처음부터 다시 시작해야하며 새로 시작할 때마다 많은 요소가 랜덤하게 달라지기 때문에 상당히 어렵기로 유명한 반면 이와 같은 특수성 때문에 게이머들에게는 인기 있는 장르이기도 하다. 그러나 이와 같은 조건을 만족시키기 위해서 게임 디자인의 관점에서는 매 게임마다 새로운 자극을 주거나 그를 위한 무수한 콘텐츠들을 마련해 놓아야 하는데 이와 같은 방법은 현실적으로 쉽지 않기 때문에 많은 게임 개발자들은 절차적 콘텐츠 생성(PCG, Procedural Content Generation)기법을 이용하여 위의 욕구를 달성시키고 있다. 절차적 콘텐츠 생성은 게임 내에서 필요한 콘텐츠가 개발자에 의해서가 아닌 특정한 알고리즘에 의해 자동적으로 생산되는 것으로 정의된다[2]-[4]. 이와 같은 절차적 콘텐츠는 게임 내 몇몇 수치들을 난수 화 시키는 것과 같이 간단한 작업만으로도 만들어 낼 수도 있지만, 이렇게 생성된 콘텐츠들은 대체로 품질에 대한 이슈를 겪게 된다[5][6].

본 논문에서는 PCG를 이용하여 로그 라이크 게임용 맵을 생성하지만 맵 내 템플릿을 추가하여 복잡도를 증가시킴으로써 품질 및 랜덤성을 향상시킬 수 있는 기법을 제안한다. Ⅱ장에서는 PCG에 관한 관련연구를, Ⅲ장에서는 템플릿을 이용한 트리구조 확장 방식을, Ⅳ장에서는 실험결과를, Ⅴ장에서는 결론에 대해 설명한다.

Ⅱ. 관련 연구

최근에 부각되는 콘텐츠 생성 기법은 게임 내에서 필요한 콘텐츠가 개발자에 의해서가 아닌 특정한 알고리즘에 의해 자동적으로 생산되는 것으로 정의된다. 이를 통해 매번 새로운 맵을 디자인하는 시간 및 인력이 절약되어 게임을 수행할 때마다 다른 맵이 요구되는 로그 라이크 게임 장르에서는 핵심적인 알고리즘이다. PCG에 의해 맵을 생성하는 방법에는 일반적으로 BSP(Binary Space Partitioning)와 트리구조 확장 방식이 있다[7].

2.1 BSP

이진 공간분할법은 하나의 공간을 최종목적에 도달할 때 까지 재귀적으로 2개씩 분할하는 과정이다, 해당 방법론은 이진 공간분할법을 통해 생성된 완전히 분리된 공간들을 활용하여 절차적 미로를 생성한다. 그 일련의 과정은 변수할당, 공간분할, 방생성 및 통로연결의 과정을 이루어진다.

첫째, 변수할당과정에서는 공간을 나누는데 사용될 난수의 범위를 결정한다. 이때 난수의 범위 폭이 넓을수록 과하게 작거나 큰방이 생성될 수 있으며, 좁을수록 매번 비슷한 형태의 맵이 생성된다. 둘째, 공간분할 과정에서는 맵을 저장할 전체 공간을 하나의 노드로 만들고 트리의 새로운 잎으로써 기록한다[8]. 이때 노드는 해당공간의 좌 하단 좌표와 우 상단 좌표 값을 정보로 갖는다. 그리고 범위가 결정된 난수를 생성하여, 해당 비율에 맞춰 주어진 공간을 수평 혹은 수직으로 분할한다. 이때 분할로 만들어진 두 개의 공간은 별개의 노드로서 기록하며 기록되어 있는 노드의 수만큼 새로 난수를 생성하여 분할한다. 이때 분할 방향은 수직방향과 수평방향을 진행되며 이전의 분할방향과 반대로 분할을 진행한다. 만들어진 공간들은 별개의 노드로서 새로 기록한다. 반복 횟수가 N보다 작다면, 일련의 과정을 반복한다.

셋째, 방생성에서는 최종적으로 기록된 노드들로부터 분할된 공간의 정보를 받아온다. 그리고 공간을 일정한 비율만큼 차지하는 방을 생성해준다. 이때 방의 크기가 목표로 하는 방의 최소크기를 만족하지 못한 경우 비율을 새로 산정하여 방을 다시 생성한다.

마지막 통로 연결하기에서는 최종적으로 나누어진 방으로부터 다시 트리를 거슬러 올라가며 통로를 이어 미로를 완성한다. 그림 1은 위에서 설명한 과정을 거쳐 생성된 맵을 나타낸다. 그림 1 내 노란색 표시선은 시작점에서 최종 탈출구까지의 경로를 나타낸다.

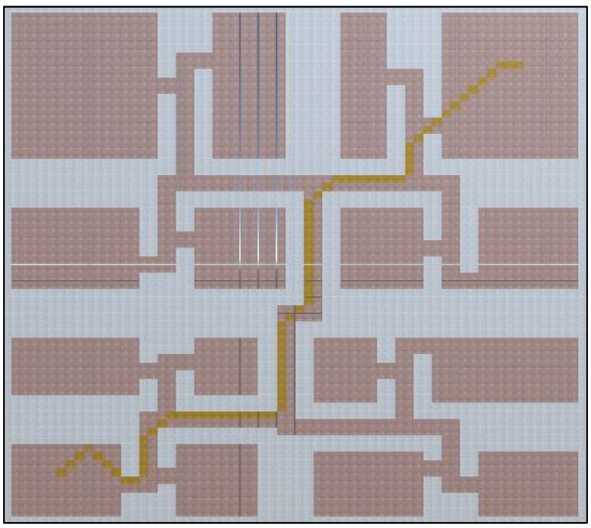


그림 1. BSP에 의해 생성된 경로
Fig. 1. Path generation by BSP

그림 1에서와 같이 BSP를 활용한 절차적 콘텐츠 생성방법은 알고리즘에 의해 맵이나 미로생성이 용이하여 개발자의 수고를 덜 수는 있다. 그러나 공간을 나누는 난수의 비율설정에서 난수 범위를 넓히면 잘못된 결과 값이 나올 수 있기에 분할비율을 크게 바꿀 수 없어 생성된 미로들은 일정한 틀을 크게 벗어나지 못하여 품질이 다소 떨어진다는 단점을 가지며 맵의 난이도나 복잡도 또한 그다지 높지 않다.

2.2. 트리구조 확장 방식

트리구조 확장 방식은 알고리즘의 명칭이 따로 있는 것은 아니지만, 간단하게 구현할 수 있는 만큼 절차적 맵 생성의 한 방법으로 많이 사용되어왔다. 해당 방법론은 BSP처럼 방을 만들어낼 공간을 미리 확보해 놓기보다는, 해당 공간에 방을 생성할 수 있는지 조건만을 판별하여 계속해서 방을 확장시킨다. 몇 개가 만들어질지 알 수 없다는 단점을 갖는다. 알고리즘은 임의좌표생성, 방생성의 과정으로 이루어진다.

첫째, 임의 좌표 생성과정에서 미로를 생성할 공간의 중심점에서 일정한 가중치를 줘서 임의의 좌표를 생성한다. 생성한 좌표를 기준으로 상하좌우 중, 임의의 방향 값을 받아 방 생성을 시작한다.

둘째, 방 생성과정에서 방의 최대 크기와 최소 크기를 기반으로 생성할 방의 크기를 산정한다. 이 때 할당받은 좌표를 기준으로 방의 크기만큼 공간이 비어있지 않다면 그대로 종료한다. 하지만 만약 공간이 비어있다면 새롭게 방을 생성해주고, 방의 상하좌우에 문을 생성하여 '방 생성'을 반복한다. 그림 2는 위의 설명과 같이 생성된 맵을 나타낸다. 그림 2내의 초록색 표시선은 시작점에서 최종 탈출구까지의 경로를 나타낸다. 그림 1과 그림 2를 비교하였을 때 트리구조 확장 방식에 의해 얻어진 그림 2의 결과에서 방의크기가 다양하게 얻어졌음을 알 수 있다.

BSP방식에 의해 얻어진 그림 2의 결과에 비해 맵의 랜덤성이나 복잡도가 증가함을 보이며 변화의 폭이 넓음을 알 수 있다. 그러나 그림 4에서와 같이 방의 모양이 직사각형으로만 되어있는 방만을 반환함으로써 게임 플레이어에게 흥미를 유발시킬 수 없으며 전체적인 맵의 품질이 떨어지며 랜덤성을 충족하기 어렵다[9]. 따라서 방의구조를 변화시키기 위해 템플릿 도입을 제안한다.

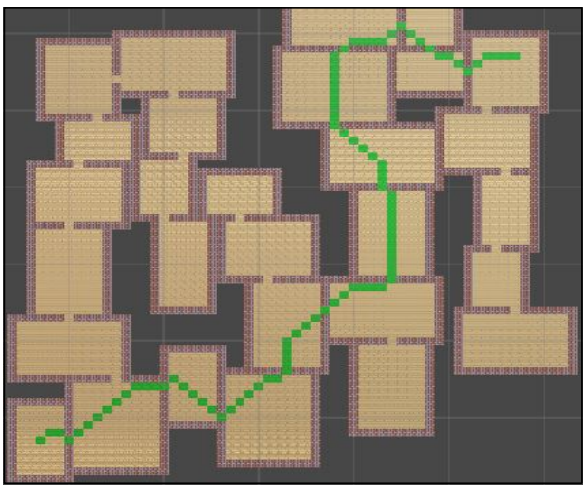


그림 2. 트리구조 확장 방식에 의해 생성된 경로
Fig. 2. Path generation by extended tree structure

III. 템플릿을 이용한 트리구조 확장 방식

2장에서 설명된 트리구조 확장 방식이 BSP방식보다 좀 더 랜덤하고 복잡도가 증가되었지만 방의 구조가 직사각형으로 생성되어 랜덤성이나 복잡도가 다소 미흡하다는 단점이 있었다. 이와 같은 문제점을 개선하기 위해 본 논문에서는 트리구조 확장 방식에 템플릿을 도입하여 좀 더 다양한 모양의 방을 생성하여 게이머의 흥미를 높이고 랜덤성이나 복잡도를 향상시키도록 한다.

3.1 템플릿 생성 및 검증

템플릿은 트리구조 확장 방식에의 해 생성된 직사각형구조의 방에 대신 넣어줄 다양한 형태의 구조를 의미한다. 그림 3은 다양한 형태의 템플릿을 포함한 전체 방의구조 즉 맵의 구조를 나타낸다.

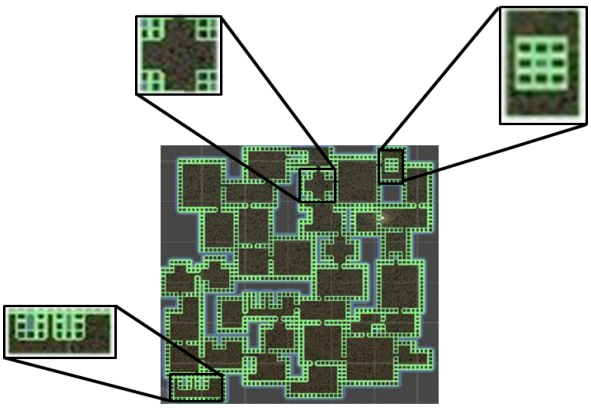


그림 3. 다양한 템플릿이 적용된 맵의 구조
Fig. 3. Map with different templates

그림 3에서와 같이 본 논문에서 사용할 템플릿은 사각형을 기반으로 변형시킨 구조를 함으로써 새로운 알고리즘을 만들어야 하는 어려움을 극복함과 동시에 과도한 연산을 수행하지 않아도 충분히 랜덤성과 복잡성을 향상시키는 장점을 갖는다. 템플릿을 생성하는 과정은 다음과 같다.

1) 먼저 템플릿을 적용시킬 사각형 형태의 방(템플릿 적용방)이 있다고 가정한다.
2) 해당 템플릿 적용방의 크기보다 작은 또 하나의 사각형을 무작위로 생성한다.
3) 사각형을 템플릿 적용방 내 무작위 위치에 배치한다.
4) 사각형 크기만큼 템플릿 적용방의 공간을 삭제한다.
5) 삭제된 공간과 맞닿는 템플릿 적용방의 경계에 벽을 생성해 준다.

그림 4는 최종 생성된 템플릿을 타나낸다. 그림 4에서 검정영역은 상기 두 번째 과정에서 생성된 사각형만큼 비워진 공간을 의미하며 초기 템플릿적용방에서 공간이 제거되고 만들어진 템플릿은 ㄱ자 모양을 하게 됨을 알 수 있다. 두 번째 과정에서 만들어지는 사각형의 가로세로 크기에 따라 다양한 ㄱ자 모양이 생성됨을 알 수 있다.

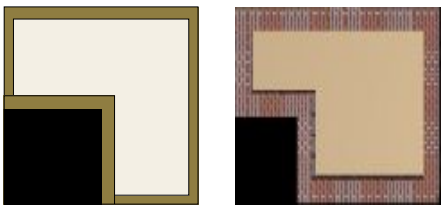


그림 4. 다양한 구조를 갖는 템플릿
Fig. 4. Templates with different size

그림 4와 같이 템플릿이 생성되면 생성된 템플릿이 제대로 된 방인가 즉 유효템플릿인가를 검증하기 위한 검증과정이 다음과 같이 진행된다.

1) 먼저 방에서 '삭제된 공간'과 '벽'을 제외한 모든 좌표를 '이동 가능 지점'이라고 가정한다.
2) '이동 가능 지점'중 한 곳을 무작위로 선정하여, 그 위치를 A라고 한다.
3) A에서 다른 모든 '이동 가능 지점'으로 이동 가능한지를 검사한다. 검사알고리즘은 대각선 이동이 제한된 A* 알고리즘을 사용한다.
4) 3)의 과정에서 만약 이동이 불가능한 경우의 수가 있다면 해당 방은 완전하게 연결되어 있지 않은 것이므로 하나의 방이라 할 수 없으며 따라서 유효템플릿이 아닌 것으로 판별한다.
5) 3)의 과정에서 모든 경우의 수를 만족시킨다면, 해당 방은 완전하게 연결되어 있는 방으로 유효템플릿으로 판별한다.

이와 같은 템플릿생성 및 검증과정을 거쳐 기본적으로 만들어질 수 있는 방의 크기에 맞는 다양한

크기의 유효템플릿을 만들어 데이터베이스에 구조체의 배열형태로 저장한다.

3.2 템플릿 적용

템플릿을 적용하기 위해서는 기존의 트리구조 확장 방식에서처럼 맵이 만들어질 공간의 중심에 방을 생성하는 것이 먼저 진행된다. 트리구조 확장 방식의 결과로 사각형 형태의 방이 만들어지면, 그 방의 크기와 같은 템플릿들을 '후보 템플릿'으로 두고 그 중 하나를 랜덤 선택하여 맵에 적용시킴으로써 다양한 크기와 형태를 갖는 방을 갖는 맵을 구성할 수 있다. 템플릿의 선택은 맵 다양성을 위해 우선순위 없이 랜덤하게 선택하지만 선택된 템플릿 적용 시 다른 방들과 연결되지 않은 방이 만들어 질 수 있으므로 템플릿을 적용시킨 전체 방 즉 맵의 구조가 이 적절하게 생성되었는가를 평가한다. 적절하다고 판별된 경우 그대로 템플릿을 적용시키며, 만약 적절하지 않다고 판별된 경우 '후보 템플릿'에서 적절하지 않다고 판별된 템플릿을 제외하고 다시 랜덤선택 하여 재판별 한다. 모든 후보가 탈락하여 '후보 템플릿'이 비게 될 경우 원래의 사각형 형태의 방을 반환하여 방의 생성을 해당 방의 생성을 마무리하게 된다. 그림 5는 템플릿이 적용된 맵생성 결과를 나타낸다. 그림 1또는 그림 2에 비해 다양한 크기와 형태의 방들로 구성된 맵이 생성되었음을 알 수 있다.

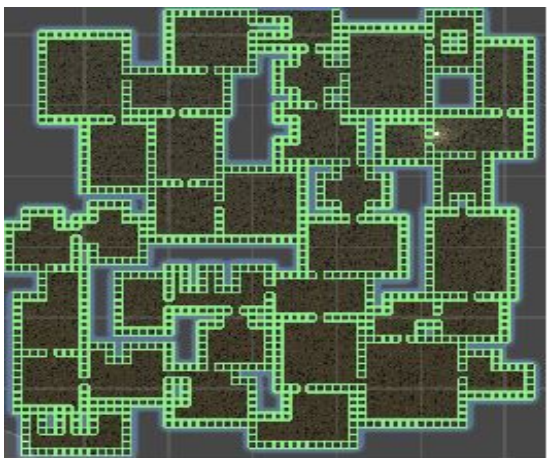
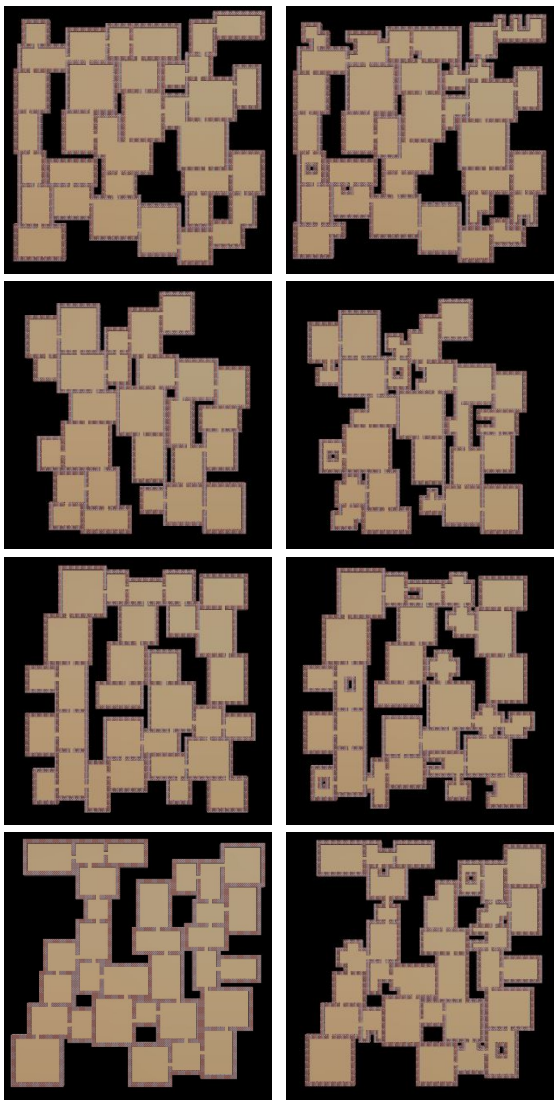


그림 5. 템플릿이 적용된 맵
Fig. 5. Map with templates

Ⅳ. 결과 및 분석

본 논문에서는 맵의 복잡도를 증가시켜 게임의 흥미를 높이기 위해 템플릿을 이용한 방의 구조 변형방법을 제안하였다. 그림 6은 템플릿 적용 전·후의 맵의 구조를 나타낸다. 그림 6(a)는 템플릿적용 전의 맵의 구조로 트리구조 확장 방식에 의해 생성된 맵을 나타낸다. 그림 6(b)는 템플릿 적용 후의 맵구조를 나타내며 그림 6(a)에 비해 다양한 형태의 방이 포함됨을 볼 수 있다.



(a) 템플릿 적용 전 (b) 템플릿 적용 후

그림 6. 템플릿 적용 전·후의 맵
Fig. 6. Map of before and after application of templates, (a) Before application of templates, (b) After application of templates

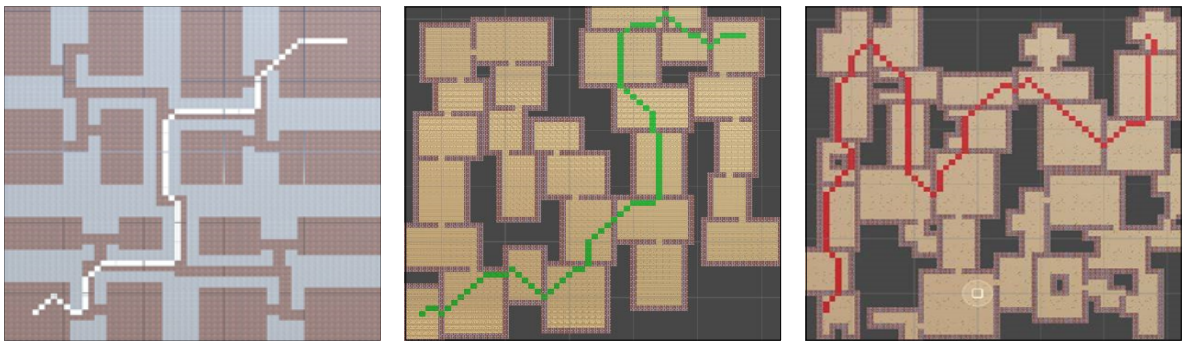
그림 7은 2장, 3장에서 설명한 절차적 생성기법인 세 가지 알고리즘에 의해 생성된 맵에서 방을 탈출하기 위한 시작위치에서 최종 위치까지의 이동경로를 비교한 그림이다.

그림 7(a)는 이진공간분할법인 BSP알고리즘에 의해 분할된 맵을 나타내며 방을 탈출하기 위한 시작위치와 마지막위치를 잇는 이동경로는 흰색으로 표시하였다. 그림 7(b)는 트리구조 확장 방식 알고리즘에 의해 분할된 맵을 나타내며 방을 탈출하기 위한 시작위치와 마지막위치를 잇는 이동경로는 녹색으로 표시하였다. 그림 7(c)는 템플릿을 이용한 트리구조 확장 방식 알고리즘에 의해 분할된 맵을 나타내며 방을 탈출하기 위한 시작 위치와 마지막위치를 잇는 이동경로는 적색으로 표시하였다. 색깔별로 표시된 이동경로는 생성된 맵에서 가장 왼쪽 아래에 위치한 방과 가장 오른쪽 위에 위치한 방을 선정한 후 빠른 경로 찾기 알고리즘을 이용하여 두 개 방사이의 최단 경로를 구한 것이며, 구해진 최단경로를 각각 흰색, 녹색 및 적색으로 표기한 것이다.

본 논문에서는 맵의 랜덤성 및 복잡도 증가를 비교하기 위해 각 알고리즘별로 구해진 최단경로 내 노드의 개수 및 방향전환횟수를 구하였다. 통과해야 할 노드 개수 및 방향전환횟수의 증가는 맵 내 복잡도 및 랜덤성이 증가됨을 의미 한다. 여기에서 노드는 이동경로 내 포함된 좌표를 의미하며 64×64의 맵인 경우 그 수만큼의 노드가 있다는 의미이다. 방향전환횟수는 8방위를 기준으로 최단거리 내 노드간 이동 시 방향의 변화를 나타내는 지수이다.

표 1은 각 알고리즘별로 10회 생성된 맵 내 시작위치에서 최종위치까지 통과한 노드의 개수를 표시한 것이다. 표 1에서 BSP알고리즘은 10회 평균 77.5개의 노드를 통과하였으며, 트리구조 확장 알고리즘의 경우 평균 91.1개의 노드를, 제안된 방법인 템플릿적용 트리구조 확장 방식의 경우 99.3개의 노드를 통과함을 알 수 있다. 따라서 제안된 방법인 템플릿 이용 트리구조 확장 방식이 BSP방식에 비해 평균 17.5%, 트리구조 확장 방식에 비해 9% 증가됨을 알 수 있다.

표 2는 각 알고리즘별로 10회 생성된 맵 내 시작위치에서 최종위치까지 최단 이동경로 내에서 발생한 방향전환횟수를 표시한 것이다.



(a) 이진공간분할방법 (b) 트리구조 확장 방식 (c) 템플릿을 이용한 트리 구조 확장 방식

그림 7. 템플릿 적용 전・후의 맵

Fig. 7. Map of before and after application of templates, (a) BSP, (b) Extended tree structure, (c) Extended tree structure using templates

표 1. 시작위치에서 최종위치까지 이동경로 내 노드의 개수

Table 1. Number of nodes between start point and last point

Number of moves	1	2	3	4	5	6	7	8	9	10	Average
BSP	80	82	73	83	75	75	80	75	73	79	77.5
Extended tree structure	89	144	82	82	78	104	67	85	107	76	91.1
Extended tree structure using templates	83	96	138	74	110	139	64	81	128	80	99.3

표 2. 시작위치에서 최종위치까지 이동경로 내 방향전환 횟수
Table 2. Number of direction transition in path between start point and last point

Number of direction transition	1	2	3	4	5	6	7	8	9	10	Average
BSP	20	17	16	14	14	14	14	16	12	14	15.1
Extended tree structure	29	28	18	25	20	25	19	23	24	20	23.1
Extended tree structure using templates	24	26	45	23	25	40	17	22	30	27	27.9

표 2에서 BSP알고리즘은 10회 평균 15.1회의 방향전환이 있었으며, 트리구조 확장 알고리즘에서는 평균 23.1회, 제안된 방법에서는 평균 27.9회의 방향전환이 이루어졌음을 알 수 있다.

제안된 알고리즘의 경우 BSP에 비해 평균 85%, 트리구조 확장 방식에 비해 20%이상 더 많은 방향전환이 있었음을 알 수 있다. 따라서 제안된 방법이 기존의 다른 방법에 비해 맵의 랜덤성 및 복잡도가 향상됨을 알 수 있다.

Ⅳ. 결 론

본 논문에서는 로그 라이크 게임에서 용이하게 변화가 이루어지는 맵을 개발하기 위한 개발자와 지속적인 재미를 요구하는 게이머의 요구를 모두 만족할 수 있는 맵생성기법을 제안하였다.

기존의 절차적 콘텐스 생성기법을 이용하여 게임을 개발하지만 사용되는 맵을 생성하기 위해 기존 절차적 콘탠츠 생성기법으로 만들어진 맵에 템플릿을 이용한 트리구조 확장 방식을 적용한 맵생성기법을 제안하였다. 템플릿은 정형화된 사각형 방의 구조를 다양한 형태 또는 다양한 크기의 방으로 변형시킴으로써 복잡도 및 랜덤성을 증가시켜 게임개발자의 입장에서는 절차적 컨텐츠생성기법(PCG)인 트리구조 확장 방식을 이용하여 게임컨텐츠를 자동으로 생산하여 짧은 시간 내 많은 양의 컨텐츠를 만들 수 있게 하며 게이머 입장에서는 매 게임마다 새로운 경험을 할 수 있도록 랜덤성과 복잡성을 증가시키되 PCG방식개발에서 나타나는 품질저하문제를 해결할 수 있었다. 현재 개발자에게는 편의성을 제공하며 게임 유저에게는 고품질, 높은 랜덤성을 갖는 게임을 개발하기 위해 지속적인 연구를 진행중이다.

References

[1] Forde, Jack, "The Evolution of the Roguelike", IGN. Retrieved Jan. 22, 2016.

[2] Julian Togelius, Emil Kastbjerg, David Schedl, and Georgios N. Yannakakis, "What is Procedural Content Generation? Mario on the borderline", Proceedings of the 2nd International Workshop on Procedural Content Generation in Games, Bordeaux, France, Article No. 3, pp. 1-5, Jun. 2011.

[3] Gang Sin-Jin "Procedural Content Generation in On-line game", ACM Transactions on Multimedia Computing, Communications, and Applications, Vol. 9, No. 1, pp. 1-5, Feb. 2013.

[4] Shaker, Noor, Togelius, Julian, Nelson, and J. Mark, "Procedural Content Generation in Games", Springer International Publishing Switzerland, 2016.

[5] Ji-Mim Kim, Pyung Oh, Seon-Jung Kim, and Seok-Min Hong, "Automatic Map Generation without an Isolated Cave Using Cell Automata Enhanced by Binary Space Partitioning", Journal of Korea Game Society, Vol. 16, No. 6, pp. 59-68, Dec. 2016.

[6] Ji-Mim Kim, "Procedural game content generation using genetic algorithm and cell automata", HanLim University, Master's thesis, 2017.

[7] H. Radha, M. Vetterli, and R. Leoonardi, "Image Compression Using Binary Space Partitioning Trees", IEEE Transactions on Image Processing, Vol. 5, No. 12, pp. 1610-1624, Dec. 1996.

[8] Y. D. Youn, Y. H. Jang, K. J. Phy, K. S. Cho, J. H. Ahn, and H. Min "A Beacon-based Space Partition Scheme for Patient Location Tracking", Journal of IIBC, Vol. 18, No. 2, pp. 157-162,

Apr. 2018,
[9] C. S. Oh, "A Improved Equivalent Table Algorithm for Connected Region Labeling", Journal of IIBC, Vol. 19, No. 1, pp. 261-264, Feb. 2019.

저자소개

김 재 현 (Jaeo-Hyun Kim)

2014년 3월 : 공주대학교 컴퓨터공학부 소프트웨어전공 입학
2019년 2월 : 공주대학교 컴퓨터공학부 소프트웨어전공 졸업(공학사)
관심분야 : 컴퓨터그래픽, 가상현실, 게임프로그래밍, 증강현실

이 부 형 (Boo-Hyung Lee)

1983년 2월 : 숭실대학교 전자공학과(공학사)
1989년 8월 : 숭실대학교 전자공학과(공학석사)
1998년 2월 : 숭실대학교 전자공학과(공학박사)
2021년 1월 현재 : 공주대학교 컴퓨터공학부 교수
관심분야 : 실시간 영상처리, 컴퓨터비젼, 물체인식, 증강현실