Journal of Korea Game Society 2015 Aug; 15(4): 157-168
http://dx.doi.org/10.7583/JKGS.2015.15.4.157

게임 프로그래밍

경로 탐색 알고리즘을 이용한 게임 레벨 난이도 평가

전영재, 오경수
숭실대학교 미디어학과
{dkreformer, oks}@ssu.ac.kr

Difficulty Evaluation of Game Levels using A Path-Finding Algorithm

Youngjae Chun, Kyoungsu Oh
Dept. of Media, Soongsil university

요 약

게임의 난이도는 게임의 재미와 깊은 연관이 있다. 하지만 게임 레벨의 난이도를 적절하게 결정하는 것은 쉽지 않다. 대부분의 경우 사람의 실제 게임 플레이를 통한 테스트가 요구한다. 또한 정량적인 평가도 어렵다. 따라서 게임 레벨 난이도의 정량적 평가를 자동으로 수행하는 것은 게임 개발에 많은 도움이 될 것이다. 이 논문에서는 경로 탐색 알고리즘을 사용하여 게임 레벨의 길 찾기 난이도를 평가하였다. 길을 찾는 것은 많은 게임들의 기본 속성으로 게임 레벨의 전반적인 난이도를 대표한다. 그리고 우리는 게임 레벨의 탐색 가능 영역이 동적으로 확장되고 다시금 탐색이 요구되는 경우 이전 경로 탐색 결과를 재사용하여 난이도 평가 알고리즘의 성능을 최적화하였다.

ABSTRACT

The difficulty of the game is closely related to the fun of the game. However, it is not easy to determine the appropriate level of difficulty of the game. In most cases, human playtesting is required. But even so, it is still hard to quantitatively evaluate difficulty of the game. Thus, if we perform quantitative evaluation of the difficulty automatically it will be very helpful in game developments. In this paper, we use a path finding algorithm to evaluate difficulty of exploration in a game level. Exploration is a basic attribute in common video games and it represents the overall difficulty of the game level. We also optimize the proposed evaluation algorithm by using previous exploration histories when available area in an game level is dynamically expanded and the new search is required.

Keywords : Difficulty evaluation, Path finding algorithm, Game level

Received: July, 10, 2015 Accepted: Aug, 05, 2015
Corresponding Author: Kyoungsu Oh(Soongsil university)
Email: oks@ssu.ac.kr

ISSN: 1598-4540

1. 서 론

게임의 점진적인 난이도의 상승은 지속적인 게임 플레이를 유도하여 재미를 느끼도록 만드는 중요한 요소이며 게임의 성공 여부에 많은 영향을 끼친다. 하지만 일반적으로 다양한 단계의 난이도를 갖는 게임을 만드는 데엔 많은 시간과 노력이 요구된다. 그리고 게임 레벨의 난이도는 실제로 플레이 하지 않으면 평가하기 힘든 경우가 많다. 따라서 게임 레벨의 난이도 평가를 자동으로 수행하는 것은 보다 양적, 질적인 측면에서 효율적인 게임 개발을 위해서 필수적이다. 또한 최근에 많이 연구되고 있는 게임 레벨 자동 생성이나 동적인 게임 난이도 조절을 위해서는 게임 레벨에 대한 난이도 평가가 선행되어야 한다.

이 논문에서 우리는 게임이 일반적으로 포함하고 있는 길 찾기에 집중한 난이도 평가 방법을 소개한다. 또한 기존의 길 찾기 결과를 재사용하여 제안하는 방법을 최적화하였다. 길 찾기는 게임의 구성 요소 중 가장 기본적인 속성으로 이를 통해 난이도를 평가하는 것은 의미가 있다. 다양한 요소를 포함하고 있는 게임의 난이도를 종합적으로 평가한다는 것은 매우 어려운 일이다. 따라서 기존의 난이도 평가 방법들은 목표 게임의 특성들을 고려하여 난이도 평가를 수행하였으나 이는 제한적이었다[1,2,3,4,5]. 우리는 게임의 일반적인 특성인 길 찾기에 대한 난이도를 평가하여 전반적인 난이도를 추정하고자 하였다. 이를 위해서 잘 알려진 경로 탐색 알고리즘인 A* 알고리즘을 사용하였다. 그 결과 게임 레벨의 기본적인 난이도 평가가 가능하였으며 기존 탐색 결과를 재사용하여 난이도 평가 방법을 최적화할 수 있었다. 우리는 탐색이 가능 영역을 제한한 후 점차 확장해 나가는 형태의 게임이 많다는 점에 착안하여 난이도 평가 방법을 최적화하였다. 제안하는 평가 방법은 게임을 진행함에 따라 탐색 가능한 영역이 확장되고 새로운 경로 탐색이 필요하면 이전 탐색 결과를 재사용하였다. 또한 최적화된 알고리즘은 이전 탐색 도중 막다른 길이 발견되면 그 이후 해당 위치로의 탐색은 다시 수행하지 않았다. 제안하는 방법의 확장을 위해서는 길 찾기 이외의 특징들에 대한 평가 방법과의 조합을 고려할 수 있다.

이 논문의 2장에서는 기존의 게임 레벨 평가와 제안하는 방법과 관련된 경로 탐색 알고리즘들에 대해서 다룬다. 3장에서는 탐색에 기반을 둔 게임 레벨을 정의하고 4장에서는 경로 탐색을 통한 게임 레벨의 난이도 평가 방법과 최적화 방법을 설명한다. 5장에서는 실험을 통해 제안하는 방법을 검증하고 마지막으로 6장에서는 이 논문의 방법이 기여하는 바와 향후 연구에 대해 다룬다.

2. 관 련 연 구

2.1 게임 레벨 난이도 평가

게임 개발의 난이도 평가는 특정한 모양과 규칙을 갖는 게임 레벨의 난이도를 수치적으로 나타냄으로써 상대적인 난이도 차이를 알게 해준다. 많은 연구들은 게임 레벨을 자동으로 생성하기 위해 또는 점차 어려워지는 난이도의 게임 레벨을 순차적으로 제공하기 위해 게임 레벨 난이도 평가를 수행한다.

슈퍼마리오(Super Mario Bros.)는 대표적인 2차원 횡방향 진행 액션 게임[6]으로 이 게임이 가지는 특징을 토대로 게임 레벨을 자동 생성하고자 하는 연구들이 많이 있었다[1,2,3]. [1]은 게임 레벨을 구성하는 주요 구성 요소를 6개로 정의하고 유전 알고리즘을 사용하여 게임 레벨을 생성하였다. 구성 요소 중 뛰어 넘어야 하는 틈이나 적 캐릭터는 사용자에 대한 도전적 요소로 간주된다. 이런 도전적 요소들에 대한 난이도 평가 합산이 곧 게임 레벨의 전체 난이도이며 이는 유망한 유전자 선택을 위한 적합도 함수에서 사용된다. 하지만 방법이 너무 간단하며 2차원 횡방향 진행 액션 게임에만 적용 가능한 방법인 문제점이 있다. [2]는 기존의 방법을 보다 확장하여 유전 알고리즘의 적합도 함수로 사용될 수 있는 난이도 평가 모델을 제

안하였다. 이 모델은 게임 레벨의 재미를 최대화 하는 모델로써 기존의 방법[1]에서 제안한 난이도 평가 방법에 사용자의 분노를 측정하는 모델과 몰입에 대한 모델을 추가하였다. 하지만 여전히 적용 가능한 게임 방식에 제한이 있었다.

[3]은 사용자에게 적합한 난이도의 게임 레벨을 제공하기 위한 연구를 진행하였다. 제안하는 방법은 사용자에 따라서 자동으로 횡방향 액션 게임을 만들어주는 것이다. 이를 위해서 사용자의 플레이 방법이나 행동 결과를 통계학적으로 분석하였다. 그 결과 사용자가 게임을 플레이 동안 얼마나 재미를 느끼는지를 추정하고 이에 맞추어 적합한 콘텐츠를 제공하였다.

[4]는 게임 규칙을 진화시켜 다양한 규칙과 형태의 보드 게임을 자동으로 생성하였다. 생성한 보드 게임들에 대한 적합성 평가를 위해서는 인공지능을 사용하여 가상의 대전을 수행하였다. 미리 정의된 57 개의 게임 디자인 규칙은 가상 대전에 대해 점수를 계산하며 결과적으로 모든 플레이어에게 공정하면서도 드라마틱한 플레이가 가능한 최적의 게임 규칙을 만들어낸다. 다만 보드 게임의 난이도는 인공지능에 의존적이며 단계적으로 보드 게임을 제공하는 연구는 진행되지 않았다.

[5]는 실시간 전략 게임 레벨의 자동 생성을 위한 밸런스 평가를 수행하였다. 이는 [4]가 자동 생성하는 보드 게임과 같이 2명 이상의 사용자가 게임을 같이 하는 것을 가정한다. 거리와 관련한 6개의 평가 기준은 자동 생성한 게임 레벨이 각 플레이어들에게 얼마나 공정하고 유사한 기회를 제공하는지 평가한다. 이 방법의 특징은 인공지능 플레이어를 사용하지 않고 거리와 면적 등의 정보들을 활용하여 게임 레벨을 평가한 점이다. 이 방법은 [4]의 단점과 유사한 단점을 가지고 있다.

2.2 경로 탐색 알고리즘

게임의 난이도 평가를 위해서 경로 탐색 알고리즘을 사용하는 경유는 드물다. 하지만 컴퓨터를 사용하여 경로를 탐색하도록 하고 탐색 결과를 토대로 난이도를 평가하기 위해서는 인공지능 플레이어 같은 에이전트를 활용할 필요가 있다. 또한 기존의 경로 탐색 알고리즘과 달리 게임 레벨의 탐색에 적합하도록 최적화할 여지가 있다.

A* 알고리즘은 [7]이 처음 소개한 알고리즘으로 깊이 우선 방식이나 너비 우선 탐색 방식과 달리 최고 우선 탐색을 수행한다. A* 알고리즘은 추정 탐색 비용인 휴리스틱(Heuristic)을 얼마나 잘 작성하는가에 따라 더욱 효율적인 경로 탐색이 가능하다. 컴퓨터 게임에서는 A* 알고리즘을 사용하여 논 플레이어 캐릭터(Non-Player Character) 또는 적 캐릭터 등의 이동 경로를 결정하기도 한다.

A* 알고리즘은 유용하고 유명하기 때문에 다양한 분야의 연구에서 알고리즘의 확장이 시도되었다. 특히 기본적인 A* 알고리즘은 정적인 환경에서 수행되기 때문에 동적인 환경에 적용시키지 위한 연구들이 많았다. 탐색 가능한 지점이 사라지거나 새로 생기는 등의 환경의 변화는 최적의 경로를 변화시킬 수 있다. 따라서 이런 환경의 변화가 생기는 경우 기존의 탐색 정보를 재사용하는 연구들이 주를 이룬다.

평생 계획(Lifelong Planning) 경로 탐색 알고리즘은 이전 탐색 정보를 재사용하는 방법이다[8,9]. 주어진 시작 위치와 목표 위치에 대해서 일반적인 방법으로 탐색을 하는 도중 어떤 노드 간 이동에 대한 비용이 변경되거나 이동 가능한 새로운 노드가 생기거나 혹은 사라지면 현재 탐색을 중단한다. 그리고 새로운 최적의 경로를 찾는다. 이 때 기존의 탐색 정보가 재사용되어 빠르게 최적의 경로를 찾을 수 있다. [10]은 저자들의 기존 작업[8,11]들을 조합하여 더욱 복잡한 환경에 적응적인 경로 탐색 알고리즘을 소개하였다.

[12]는 정방형 격자로 표현되는 환경의 효율적인 탐색을 위해 전처리 과정을 거쳐 막다른 길에 해당하는 격자들을 제거하였다. 하지만 전처리 과정을 거친다는 것은 환경에 대한 정보를 전부 알고 있는 상태에서 경로 탐색을 수행한다는 것이기 때문에 연구의 기여는 낮았다. 또한 동일한 환경에

대해 여러 번의 탐색을 연속적으로 수행하게 되면 막다른 길에 대한 정보의 빠른 갱신이 필요하다.

[13]은 3차원 게임에서 동적인 장애물이 많은 경우에 대한 동적 경로 탐색 방법을 소개하였다. 이 방법은 지형의 특성을 고려하여 경로 탐색 방법을 다르게 적용하는 동시에 동적인 객체와의 충돌을 피하기 위해 밀개와 끌개 방법[14]을 사용하였다. 하지만 실내와 같이 복잡한 지형을 갖는 환경에 대해서는 계층적으로 지형을 분석한 전처리 정보를 사용하기 때문에 완전 동적인 환경에 적용하기는 힘들다.

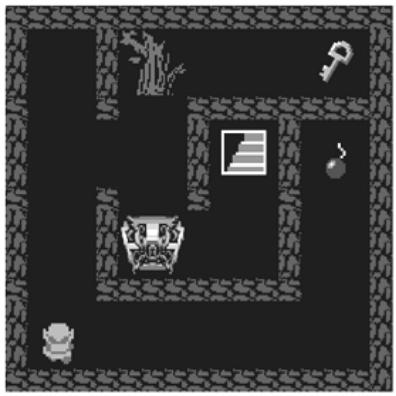
[15]는 장애물이 많은 환경에서 보다 자연스러운 탐색을 위해 사용자가 가이드라인을 제시하면 이에 맞추어 최적 경로를 탐색하는 방법을 제안하였다. 메모리 공간의 사용과 탐색 시간은 효율적이었지만 정적인 환경에만 적용 가능하고 전처리로써 가이드라인을 제공해줘야 하는 한계가 있다.

A* 알고리즘은 일반적인 경로 탐색 뿐 아니라 다양한 용도로써 활용될 수 있다. 마리오 인공지능 경진 대회(The Mario AI Competition)는 주어진 임의의 슈퍼마리오 게임 레벨에 대해서 최고의 점수를 획득하면서도 빠르게 성공적으로 게임 레벨을 완료하는 인공지능을 선출하는 경진 대회이다. [16]은 경진 대회에 참가하여 우수한 성적을 얻은 방법들과 우승한 방법을 분석하였다. 그 중 우승한 방법은 A* 알고리즘을 이용한 방법이며 게임 레벨을 동일한 크기의 2차원 격자들로 분할하여 노드를 이동할 때 마다 최적의 경로를 찾는 방법으로 우승하였다.

우리가 제안하는 방법은 개선된 A* 알고리즘들과 유사하게 이전 탐색 정보를 재사용하지만 게임 레벨의 탐색에 보다 적합하도록 최적화된 경로 탐색을 수행한다.

3. 경로 탐색 기반 게임 레벨

이 장에서는 경로 탐색 기반의 게임 레벨을 정의하고 게임 수행 방법을 설명한다. 이 게임 레벨은 처음에는 제한된 영역에 대해서 탐색을 허용한다. 그리고 사용자가 게임을 진행함에 따라 탐색 가능한 영역이 확장된다. 사용자는 수차례의 탐색을 수행한 후에 게임 레벨을 종료할 수 있다.



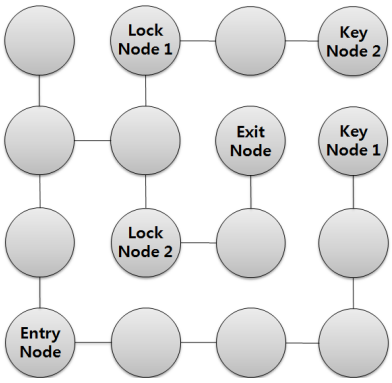
[Fig. 1] An example of game level

[Table 1] Descriptions of each node

Node	Description
Entry	A game starts at this node
Exit	The destination node of a game
Lock	This node blocks a way to the Key nodes or Exit node
Key	This node has something to unlock a particular lock node

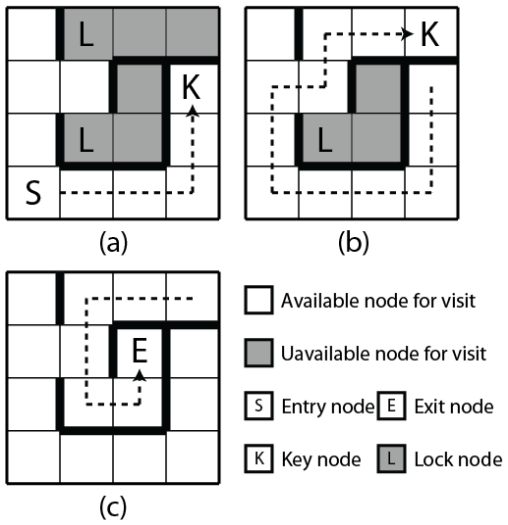
게임 레벨의 구성 요소들은 위의 [Table 1]과 같다. [Fig. 1]은 이 구성 요소들의 조합으로 만들 수 있는 게임 레벨의 예를 보여주고 있으며 [Fig. 2]는 게임 레벨을 방향성이 없는 비순환 그래프(Undirected acyclic graph)로 표현한 것이다. 많은 종류의 게임 레벨들은 다수의 노드와 연결선으로 추상화 될 수 있다. 입구 노드(Entry node)와 출구 노드(Exit node)는 각각 게임 레벨의 시작 위치와 게임 레벨의 완료를 위한 종료 위치를 나타낸다. 사용자는 출구 노드에 도달하기 위해서 게임 레벨을 탐색하여 올바른 경로를 찾아야 한다. 자물쇠 노드(Lock node)는 게임 레벨의 여러 곳에 배치되어 사용자가 탐색 가능한 노드의 개수를 제한한다. 각각의 자물쇠 노드는 하나의 열쇠 노드

(Key node)와 쌍을 이룬다. 열쇠 노드의 역할은 자물쇠 노드를 제거함으로써 탐색 가능한 영역을 확장하는 것이다. 추가된 영역을 탐색하여 최종적으로 사용자는 출구 노드에 도달할 수 있다. 우리는 게임 레벨 내의 자물쇠-열쇠 노드 쌍의 개수를 결정하는 것으로 게임 레벨의 난이도를 대략적으로 조절할 수 있다.



[Fig. 2] An example of game level represented by undirected acyclic graph

열쇠 노드는 게임 내의 도전 과제들의 일반화된 개념으로 게임 레벨 완료를 위해 해결해야 하므로 주요 도전 과제에 해당한다. 열쇠 노드에는 다양한 형태의 도전 과제가 포함될 수 있다. 단순히 방문하는 것만으로 해결되는 간단한 도전 과제에서부터 퍼즐이나 적과의 전투 등 게임의 특성과 스토리 등에 따라 결정될 수 있다. [Fig. 3]은 [Fig. 1]의 게임 레벨을 플레이하는 경우 고려할 수 있는 최적의 탐색 경로를 단계 별로 보여준다. 먼저 사용자는 (a):첫 번째 열쇠 노드에 도달하여 첫 번째 자물쇠 노드를 제거한다. 이에 의해 (b): 새로운 탐색 공간이 확장되며 사용자는 두 번째 열쇠 노드에 도달할 수 있게 된다. (c):마지막으로 두 번째 자물쇠 노드를 제거하여 출구 노드로의 경로를 확장하고 이에 도달하는 것으로 게임을 성공할 수 있다.



[Fig. 3] Overall game playing flow for an example game level

4. 게임 레벨 평가와 평가 방법 최적화

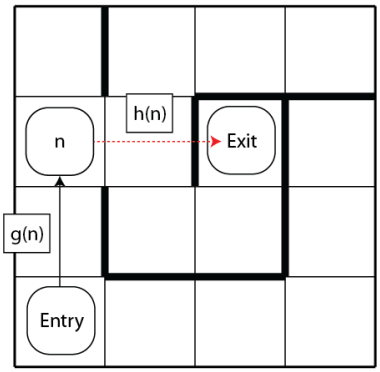
4.1 게임 레벨 탐색과 난이도 평가

본 연구에서는 게임 레벨의 탐색 난이도를 평가하기 위해 A* 알고리즘을 사용하였다[7]. A* 알고리즘은 주어진 환경과 조건에 따라서 시작 위치에서 목표 위치로의 가능한 최적의 경로를 찾는다. 우리는 게임 레벨을 탐색하는 도중에 발견되는 방문 가능한 후보 노드에 대해서 경로 탐색에 대한 비용 평가를 수행한다. 평가함수 $f(n)$은 다음과 같다.

$$f(n) = g(n) + h(n)$$

위의 식에서 n은 현재 방문하고자 하는 후보 노드를 의미하며 $g(n)$은 시작 노드에서 후보 노드까지의 경로 비용이다. 일반적으로 $g(n)$은 알려져 있지만 휴리스틱 추정 값 $h(n)$에 대해서는 알지 못한다. $h(n)$은 후보 노드에서 목표 노드까지의 추정 경로비용이며 이 추정 값을 문제 상황에 맞게 결정하는 것이 A* 알고리즘의 성능 향상에 도움을 준다. 우리는 그림 [Fig. 4]와 같이 이산적인 환경

에 자주 사용되는 맨해튼 거리(Manhattan distance)를 이용하여 휴리스틱 추정 값을 결정하였다. 하지만 제안하는 방법은 일반적인 탐색 환경에도 적용 가능하기 때문에 게임 레벨의 모양과 규칙에 따라 다른 추정 값 결정 방법을 이용할 수 있다.

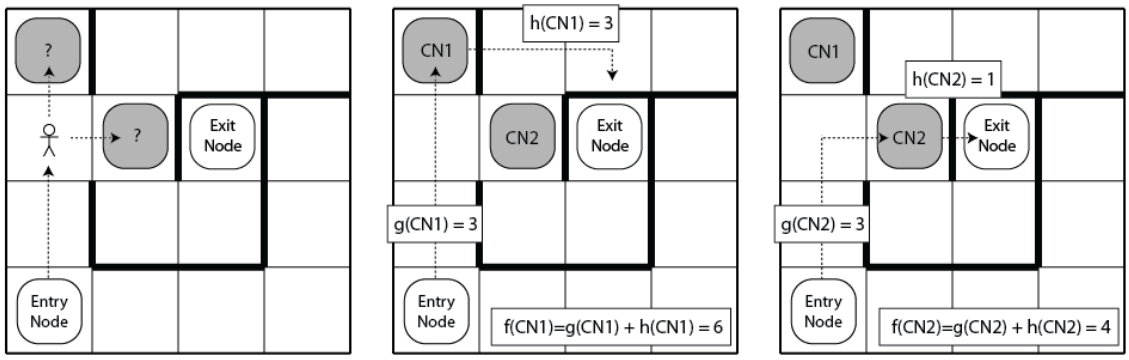


[Fig. 4] A simple example of A* algorithm

게임 레벨의 탐색은 탐색 시작 지점에서 주변 노드들을 검사하는 것으로부터 시작된다. 만약 방문 가능한 노드가 있다면 열린 목록(Open list)에 해당 노드를 저장한다. 모든 주변 노드들에 대해서 방문 가능성 여부를 조사한 후, 열린 목록에 포함된 노드들의 평가 함수 $f(n)$을 계산하여 가장 유망한 노드를 찾아서 방문을 수행한다. 그리고 직전에 방문했던 노드는 닫힌 목록(Close list)으로 옮겨서 다시는 방문하지 않도록 한다. 이와 같은 작업을 반복하면 최적의 탐색 과정을 통해 목표 위치에 도달할 수 있다.

[Fig. 5]의 왼쪽 그림은 사용자가 게임 레벨을 탐색 중 두 개의 후보 노드 $CN1$과 $CN2$를 발견한 경우이다. 그리고 가운데와 오른쪽 그림은 각 후보 노드에 대한 A* 알고리즘의 평가 함수 결과를 보여준다. 모든 노드 사이의 이동 비용을 1이라고 가정했을 때 입구 노드에서 각 후보 노드로의 이동 비용 $g(n)$은 3으로 동일하다. 이와 달리 각 후보 노드에서의 휴리스틱 추정 값 $h(CN1)$과 $h(CN2)$는 각각 3과 1로 목표 지점과의 상대적인 위치에 따라 다르다. A* 알고리즘은 $f(n)$이 더 작은 두 번째 후보 노드를 유망하다고 판단하여 다음 노드로 선택한다.

경로 탐색 알고리즘은 정해진 규칙에 따라 게임 레벨을 탐색할 뿐이지만 그 결과로 집계되는 노드 탐색 횟수는 게임 레벨의 탐색 난이도와 매우 밀접한 연관이 있다. 특히 자물쇠-열쇠 노드 쌍의 개수와 게임 레벨의 경로의 복잡함은 탐색하는 노드의 개수에 영향을 준다. 만약 자물쇠-열쇠 노드 쌍의 개수가 많다면 일반적으로 보다 많은 횟수의 탐색이 요구된다. 또한 게임 레벨에 갈림길이 많다면 탐색하는 도중에 고려해야 하는 후보 노드의 개수가 증가한다. 따라서 우리는 하나의 게임 레벨을 완전하게 종료하는 도중에 방문한 노드의 개수를 탐색 난이도로써 사용할 수 있다. A* 알고리즘을 이용하여 계산한 게임 레벨 탐색 난이도 E_{A^*}는 다음 식과 같다.



[Fig. 5] Different heuristic value of candidate node $CN1$ and $CN2$

$$E_{A^*} = \sum_{i=1}^{n+1} N_i$$

위의 식에서 n은 자물쇠-열쇠 노드 쌍의 개수로 마지막에 출구 노드로의 탐색을 1회 더하여 총 $n+1$회의 탐색에 대해 탐색한 노드 개수 N_i들을 합산한다. 각각의 N은 다음 식을 통해 구한다.

$$N = \sum_{j=1}^{k} V_j \begin{cases} 1 & \text{if } V_j is\ visited \\ 0 & otherwise, \end{cases}$$

위의 식에서 k는 전체 노드의 개수이며 V_j는 j번 노드가 탐색 도중에 방문 되었다면 1, 아니라면 0이다. 우리는 3장에서 정의한 게임 레벨들의 다양한 형태에 대해 탐색 난이도 E_{A^*}를 계산하여 특정 게임 레벨이 다른 게임 레벨에 비해서 상대적으로 어려운지 또는 쉬운지를 평가할 수 있다.

4.2 경로 탐색 알고리즘 최적화

4.1절의 방법을 사용하면 게임 레벨에 대한 탐색 난이도 평가가 가능하지만 보다 개선될 여지가 있다. 사용자가 탐색 가능한 노드의 초기 범위는 열쇠-자물쇠와 같은 게임 레벨 내 요소들로 인해 제한적이다. 하지만 게임을 진행함에 따라 [Fig. 3]의 (a), (b), (c)와 같이 점차 확장된다. 만약 이전의 탐색 정보를 활용한다면 탐색의 효율을 높이고 명백히 불필요한 탐색은 사전에 제외할 수 있다. 이것은 사람이 게임 레벨을 탐색하면서 게임 레벨의 형태와 막힌 길 등에 대한 기억을 이용하여 가능한 효율적으로 탐색을 수행하는 것과 유사하다. 우리는 다음으로 설명할 두 가지 방법을 사용하여 경로 탐색 알고리즘을 최적화하였다.

첫 번째로 직전 탐색 정보를 재활용하여 현재 탐색의 초기 정보로써 사용하였다. 이를 위해 우리는 한 번 탐색이 종료되면 열린 목록을 비우지 않고 유지하였다. 그리고 직전에 종료한 탐색에 의해 닫힌 목록에 추가 된 모든 노드들을 열린 목록으로 이동시켰다. 마지막으로 모든 열린 목록의 노드들에 대해 새로운 탐색 시작 지점과 새로운 목표 지점을 이용하여 $g(n)$과 $h(n)$을 갱신하였다. 이는 불필요한 목록 간 노드 이동과 노드 탐색을 줄임으로써 보다 효율적인 탐색이 가능하도록 한다.

두 번째로 우리는 막다른 길에 해당하는 노드와 그 노드로의 경로를 최대한 제거하였다. 만약 사람이 게임 레벨을 탐색하는 도중에 막다른 길을 발견하면 다음 탐색 시 다시 방문하지 않을 것이다. 특히 게임 레벨의 크기가 클수록 탐색을 오래 수행하게 되며 막다른 길들이 많이 발견될 확률이 높고 이 정보는 매우 유용하다.

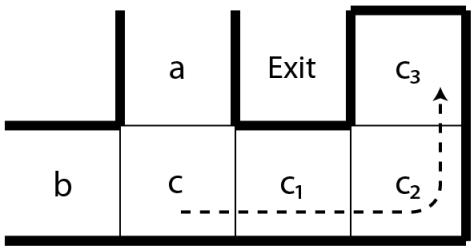
우리는 막다른 길 노드에 대해서는 휴리스틱 추정값 $h(n)$이 매우 큰 것으로 고정하여 반드시 유망하지 않은 노드로 만들었다. 또한 이런 노드들을 완벽히 닫힌 목록(Complete Close List)에 저장함으로써 재사용을 위해서도 고려하지 않았다. 마지막으로 기존의 A* 알고리즘이 탐색 도중에 더 비용이 적은 $g(n)$이 발견되면 관련된 모든 노드들의 비용 정보를 수정하는 것과 마찬가지로 막힌 길 노드가 발견되면 관련된 모든 노드들에 대한 경로 탐색 비용을 갱신하였다. [Fig. 6]은 노드 c에서 탐색을 시작하여 노드 c_1, c_2를 방문하고 막힌 길 노드 c_3에 도달한 경우를 보여준다. 먼저 막힌 길 노드 c_3에 대한 경로 탐색 비용 $f(c_3)$은 다음과 같이 갱신된다.

$$f(c_3) = g(c_3) + h(c_3),\ h(c_3) = \infty$$
$$\therefore f(c_3) = \infty$$

그리고 게임 레벨의 형태에 따라 노드 c_2을 방문한 후엔 반드시 노드 c_3을 방문하므로 경로 탐색 비용 $f(c_2)$는 다음 식을 통해 바꿔 쓸 수 있다. 이때 노드 간 이동 비용은 1로 간주한다.

$$f(c_2) = g(c_2) + h(c_2)$$
$$= g(c_2) + 1 + h(c_3),\ h(c_3) = \infty$$
$$\therefore f(c_2) = \infty$$

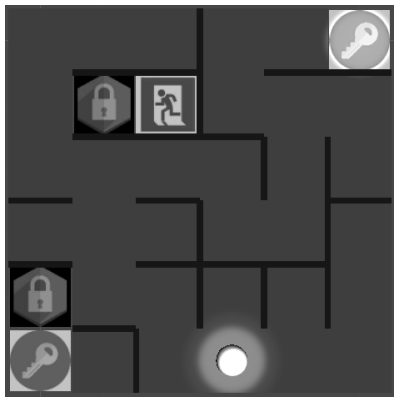
즉, 노드 c_2은 다음 노드 c_3의 휴리스틱 추정 값 $h(c_3)$으로 인해 반드시 유망하지 않게 갱신된다. 같은 과정을 통해 노드 c_1 역시 완벽히 닫힌 목록으로 이동된다. 하지만 노드 c의 경우 노드 c_1 이외에도 노드 a, b와 같이 다른 방문 가능한 노드들의 영향을 받으므로 완벽히 닫힌 목록으로 이동시키지 않는다.



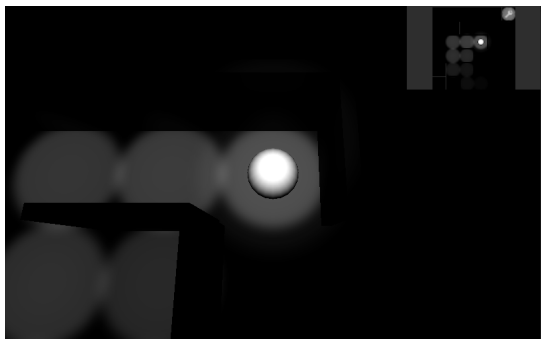
[Fig. 6] An example of dead-end finding path

5. 실험 및 결과

제안하는 게임 레벨 탐색 난이도 평가 방법은 총 노드 이동 횟수가 많을수록 상대적으로 어렵다고 판단한다. 우리는 평가 방법을 검증하기 위해 실제 사용자를 대상으로 실험을 수행하였으며 이를 위해 프로토타입을 제작하였다. 프로토타입은 3장에서 설명한 경로 탐색 기반의 게임 레벨을 제공하며 사용자는 제한된 시야 조건하에 게임 레벨을 탐색한다. [Fig. 7]은 프로토타입에서 제공하는 게임 레벨의 형태를 [Fig. 8]은 실제 사용자에게 보이는 게임 장면이다. 프로토타입은 사용자의 시야를 제한하는 대신 [Fig. 8]의 오른쪽 상단과 같이 미니맵을 표시해주며 사용자가 조종하는 공 모양의 캐릭터가 이동한 경로에 대해서 일정 시간 시야를 제공한다.



[Fig. 7] A game level



[Fig. 8] A prototype for experiments

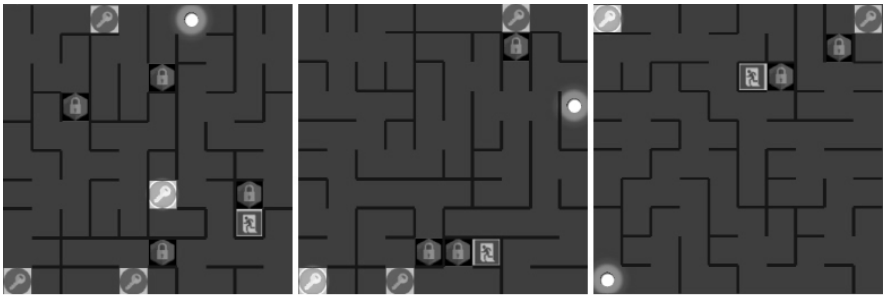
우리는 게임 레벨의 탐색 난이도를 제어하기 위해 1)게임 레벨의 크기와 2)자물쇠-열쇠 쌍의 개수를 달리하여 게임 레벨들을 작성하였다. 그리고 사용자가 다양한 게임 레벨들을 플레이 한 후 쉬운 순서대로 낮은 점수를 주도록 하였다. 이 점수의 오름차순이 난이도 평가 방법의 결과 값의 오름차순과 동일하면 제안하는 방법이 유효하다고 볼 수 있다. 우리는 게임 레벨의 크기와 자물쇠-열쇠 쌍의 개수가 각각 난이도에 영향을 줌을 검증하기 위해 게임 레벨들을 2개의 그룹으로 나누었다. 먼저 첫 번째 그룹은 2개의 자물쇠-열쇠 쌍을 포함한 3가지 크기의 게임 레벨들을 포함한다. 각각의 게임 레벨 크기는 6x6, 8x8, 10x10으로 평가 알고리즘에 의한 탐색 비용은 46, 53, 101이었다. 두 번째 그룹은 각각 2, 3, 4개의 자물쇠-열쇠 쌍을 포함한 동일한 10x10 크기의 게임 레벨들을 포함하며 탐색 비용은

101, 112, 124이었다. [Fig. 9]와 [Fig. 10]은 각 그룹별로 사용한 게임 레벨들이다. 20~30대 대학원생 남녀 7명이 실험에 참여하였으며 전공은 컴퓨터공학, 국어국문학, 법학 등으로 다양하였다.

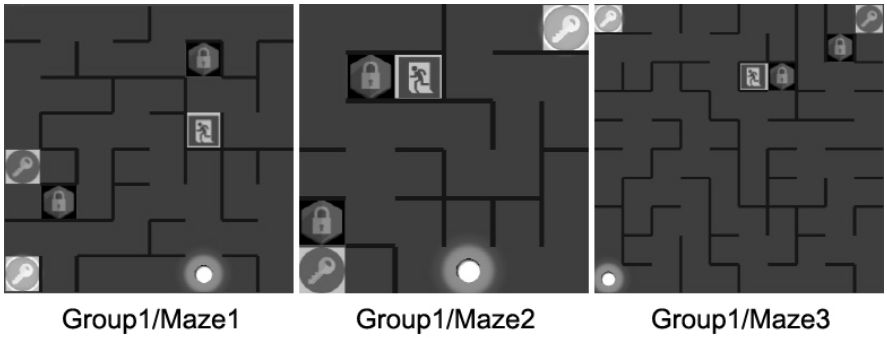
[Table 2]는 각 사용자들이 게임 레벨이 쉬운 순서대로 순번을 결정한 결과이다. G1과 G2는 두 개의 그룹을 M1, M2, M3는 각각의 게임 레벨을 의미한다[Fig. 9], [Fig. 10]). E는 제안하는 알고리즘 AI를 통해 얻은 난이도의 정량적 수치로 우리는 이 수치에 따라 게임 레벨 난이도의 순번을 결정하였다. 총 7명의 사용자(P1~P7)는 동일한 게임 레벨들을 그룹 별로 플레이 한 후 난이도에 따른 순번을 결정하였다. 이 중 약 6명이 AI의 평가 결과와 동일한 순번을 선택하였다. 따라서 제안하는 게임 레벨 탐색 난이도 평가 알고리즘은 신뢰할만하다고 볼 수 있다.

[Table 2] Human playtesting result

	G1			G2		
	M1	M2	M3	M1	M2	M3
E	53	46	101	124	112	101
AI	2	1	3	3	2	1
P1	2	1	3	3	2	1
P2	2	1	3	3	1	2
P3	3	1	2	3	2	1
P4	2	1	3	3	2	1
P5	2	1	3	3	2	1
P6	2	1	3	3	2	1
P7	2	1	3	3	2	1



[Fig. 10] Game levels in the group 2



[Fig. 9] Game levels in the group 1

6. 결 론

우리는 경로 탐색에 기반을 둔 게임 레벨을 정의하고 탐색 난이도를 평가하는 방법을 소개하였다. 제안하는 난이도 평가 방법은 최고 우선 탐색 알고리즘인 A* 알고리즘을 사용하여 탐색 비용을 기준으로 난이도를 책정하였다. 또한 게임 레벨의 탐색의 특징에 맞추어 이전 탐색 정보를 재사용하고 탐색이 불필요한 경우를 배제하여 알고리즘을 최적화하였다.

본 연구의 의의는 기존의 연구들이 시도하지 않았던 경로 탐색 기반의 게임 레벨 난이도 평가를 수행했다는데 있다. 게임의 기본적인 구성 요소인 경로 탐색은 중요하지만 겉으로 잘 드러나지 않는 경우가 많고 실제 게임 레벨의 난이도 책정에서도 고려되지 않는 경우가 많다. 특히 이런 현상은 임의로 게임 레벨의 모양을 결정짓는 형태의 게임에서 두드러진다. 제안하는 방법은 게임 레벨에 대한 탐색 난이도를 측정함으로써 게임에서의 난이도 밸런싱 또는 자동 생성 방법에서의 평가 기준으로 사용될 수 있다.

우리가 사용하는 게임 레벨은 무방향 비순환 그래프이므로 많은 형태의 게임 레벨을 표현할 수 있지만 모든 게임 레벨을 표현할 수는 없다. 향후 연구에서는 보다 일반적인 게임 레벨을 정의하고 이에 대한 탐색 난이도를 평가하기 위한 경로 탐색 알고리즘의 개선에 집중할 것이다.

ACKNOWLEDGMENTS

This work was supported by the Korea Science and Engineering Foundation(KOSEF) grant funded by the Korea government(MEST) (No.2011-0012214)

REFERENCES

[1] Sorenson, N. and Pasquier, P., “Towards a generic framework for automated video game level creation”, In Applications of Evolutionary Computation, pp. 131-140, 2010.

[2] Sorenson, N. and Pasquier, P., “The evolution of fun: Automatic level design through challenge modeling”, In Proceedings of the First International Conference on Computational Creativity, pp. 258-267, 2010.

[3] Sorenson, N., Pasquier, P. and DiPaola, S., “A generic approach to challenge modeling for the procedural creation of video game levels”, IEEE Transactions on Computational Intelligence and AI in Games, v.3, no.3, pp. 229-244. 2011.

[4] Browne, C. and Maire, F., “Evolutionary game design”, IEEE Transactions on Computational Intelligence and AI in Games, v.2, no.1, pp. 1-16. 2010.

[5] Liapis, A., Yannakakis, G. N. and Togelius, J., “Towards a Generic Method of Evaluating Game Levels”, In AIIDE, 2013.

[6] Miyamoto, S., Yamauchi, H. and Tezuka, T., “Super Mario Bros”, Nintendo, 1987.

[7] Dechter, R. and Pearl, J., “Generalized best-first search strategies and the optimality of A*”, Journal of the ACM (JACM), v.32, no.3, pp. 505-536, 1985.

[8] Koenig, S. and Likhachev, M., “Improved fast replanning for robot navigation in unknown terrain”, In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'02), v. 1, pp. 968-975, 2002.

[9] Koenig, S. and Likhachev, M., “Incremental A*”, In NIPS, pp. 1539-1546, 2001.

[10] Likhachev, M., Ferguson, D. I., Gordon, G. J., Stentz, A. and Thrun, S., “Anytime Dynamic A*: An Anytime, Replanning Algorithm”, In ICAPS, pp. 262-271, 2005.

[11] Likhachev, M., Gordon, G. J. and Thrun, S., “ARA*: Anytime A* with provable bounds on sub-optimality”, In Advances in Neural Information Processing Systems, 2003.

[12] Björnsson, Y. and Halldórsson, K., “Improved

Heuristics for Optimal Path-finding on Game Maps", In AIIDE, pp. 9-14, 2006.
[13] Oh-Ik Kwon and Teag-Keun Whangbo, "A Dynamic Path-Finding Method Avoiding Moving Obstacles in 3D Game Environment", Journal of Korea Game Society, v.6, no.3, pp.3-12, 2006.
[14] John Olsen, "Attraction and Repulsors", Microsoft, Game Programming Gems, Charles River Media, 2004.
[15] Sung Hyun Cho, "A Pathfinding Algorithm Using Path Information", Journal of Korea Game Society, v.13, no.1, pp.31-40, 2013.
[16] Togelius, J., Karakovskiy, S. and Baumgarten, R., "The 2009 mario ai competition", In Evolutionary Computation (CEC), pp. 1-8, 2010.

전 영 재(Youngjae Chun)

2007 숭실대학교 미디어학부 학사
2014 숭실대학교 미디어학부 박사
2015-현재 조지워싱턴대학교 컴퓨터공학과 박사후과정
관심분야 : 실시간 렌더링, 게임 레벨 자동 생성

오 경 수(Kyoungsu Oh)

2001 서울대학교 전기 컴퓨터 공학부 박사
2001-2002 (주)조이멘트 개발팀장
2003-현재 숭실대학교 미디어학부 교수

관심분야 : 실시간 컴퓨터 그래픽스, 시리어스 게임