유전자 알고리즘을 사용한 타워 디펜스 공격대의 자동 구성 기법※

조성현*, 강신진**
홍익대학교 게임학부
scho@hongik.ac.kr, directx@hongik.ac.kr

An Automated Wave Generation Technique in Tower Defense Games Based on a Genetic Algorithm

Sung Hyun Cho*, Shin Jin Kang**
School of Games, Hongik University

요 약

타워 디펜스 게임에서 레벨 디자인은 게임의 재미에 가장 큰 영향을 미치는 요소이다. 각 레벨의 난이도는 레벨 내에 등장하는 공격대의 조합에 따라 결정된다. 게임 기획 단계에서 게임의 재미를 주면서도 적절한 난이도를 갖춘 공격대를 구성하기 위하여 많은 시간이 소모된다. 본 논문에서는 유전자 알고리즘을 사용하여 타워 디펜스 게임에서 공격대 조합을 자동으로 생성하는 기법을 제안한다. 제안된 시스템을 통해 레벨 디자이너는 난이도 목표치의 입력만으로도 다양한 공격대 유닛 조합을 자동으로 생성할 수 있게 된다. 이는 게임 기획 단계에서 원하는 공격대 조합을 생성하는데 필요한 수작업 시간을 단축시킴으로서 업무 효율을 높일 수 있을 것이다.

ABSTRACT

Level design is one of the important factors in tower defense game development. The difficulty of tower defense game depends on its wave design. In general, it requires a lot of manual labor to generate well-balanced waves with fun. In this paper, we propose a new automated wave generation system by using a genetic algorithm. With our system, a game designer can easily generate an optimized wave by designating the difficulty level in the initial stage of game design. Our system can be useful in reducing the trial-errors in the initial level design process of tower defense game development.

Keywords : Tower Defense Game, Genetic Algorithm, Evolutionary Computation, Level Design

접수일자 : 2010년 12월 31일 심사완료 : 2011년 01월 17일
교신저자(Corresponding Author) : 강신진
※ 이 논문은 2009학년도 홍익대학교 학술연구진흥비에 의하여 지원되었음.
※ 이 논문은 2010학년도 홍익대학교 학술연구진흥비에 의하여 지원되었음.

1. 서 론

타워 디펜스 게임은 한정된 타워를 레벨에 배치하여 레벨 내에 배치된 수비 목표를 지켜내는 게임이다. 타워 디펜스 게임의 규칙은 단순하여 누구나 쉽게 게임을 플레이할 수 있지만 게임 몰입도가 높은 특징을 가지고 있다. 최근에 스마트폰 플랫폼의 어플리케이션 시장 확대와 더불어 타워 디펜스 게임은 많은 인기를 얻고 있다. Plant vs Zombie, Field Runner, Creeps, Sentinel 등이 상업적으로 큰 성공을 이룬 대표적인 타워 디펜스 게임들이다[그림 1].

[그림 1] Plant vs Zombie(상), Creeps(하)

많은 수의 타워 디펜스 게임이 존재하지만 게임의 규칙은 기본적으로 유사하다. 게임 레벨 내 수비 목표를 노리고 일정 시간 주기 마다 다양한 NPC (Non Playable Character)들로 구성된 공격대가 레벨 내 길을 따라서 침입한다. 플레이어는 공격 가능한 타워를 배치하여 공격대가 수비 목표에 도달하기 전에 제거해야 한다. 플레이어가 공격대를 제거할 때마다 일정 금액의 보상을 얻게 되고, 강한 공격대일수록 더 많은 보상을 얻게 된다. 얻어진 보상 금액으로 플레이어는 타워를 업그레이드 하거나 더 많은 타워를 레벨에 배치하는데 쓸 수 있다.

시간이 지남에 따라 NPC 공격대의 조합은 점점 강해지므로 수비 목표를 지키기 위해서 플레이어는 주어진 보상 금액을 효율적으로 사용하여 수비 목표를 지켜내야 한다. 배치된 타워와 공격대간에는 강한 상관관계가 존재한다. 플레이어가 이러한 상관관계를 잘 이해하여 타워를 업그레이드하거나 적절한 곳에 새로운 타워를 배치하는 것이 타워 디펜스 게임의 전략적 요소이다.

타워 디펜스 게임에서 공격대 디자인은 재미에 큰 영향을 미치지만 그 중요도만큼 가장 난이도가 높은 개발 업무에 속한다. 그 이유는 조합된 공격대 플레이에 대한 재미를 게임 기획 단계에서 문서상으로 정확히 예측하는 것이 불가능하기 때문이다. 결국 공격대 기획 작업 후 이를 실제로 확인하기 위하여 강도 높은 플레이 밸런싱 (Balancing) 작업이 필수적으로 이어지게 된다. 이러한 밸런싱 작업은 주관적인 영역에 속할 뿐 아니라 많은 시간이 소요되기 때문에 게임 개발팀 업무에 큰 부담으로 작용한다.

본 논문에서는 유전자 알고리즘을 사용하여, 현 레벨 디자인 작업 프로세스 상에서 레벨 디자인 구성요소의 조합에 대한 수치적 불확실성을 해소하고, 시간적 소모를 줄일 수 있는 자동화된 공격대 생성 시스템을 제안한다. 이를 통해 레벨 디자인 시에 발생할 수 있는 시행착오를 사전에 경감시킬 수 있을 것이다.

2. 관련연구

게임 레벨은 게임 시스템이 복합적으로 모두 적용되는 결과물이다. 다양한 게임 관련 기술 중 레벨 분야와 밀접하게 연관된 연구로는 인공 지능

분야로 볼 수 있다. 게임 내 레벨에 적용되는 대표적인 인공 지능 기법으로는 FSM (Finite State Machine)을 사용한 NPC 행동 제어와 A* 알고리즘과 내비게이션 메시를 사용한 길찾기 알고리즘이 있다[1,2]. 광범위한 AI 연구 분야에 비하여 게임 내 레벨에 적용되는 AI기법은 제한되어 있는데, 이는 플레이어에게 재미를 전달하기 위해 최상의 AI 기법을 선택하기 보다는 설정된 환경에서 게임 디자이너가 원하는 결과 행동을 도출할 수 있는 AI 기법을 선택하기 때문이다. 이는 제한된 시간 내에 구현 가능성이 높고 예측 가능한 재미를 전달하기 위함이다[3].

게임 레벨을 제작 하는데 많은 수작업 프로세스가 존재한다. 이는 게임 콘텐츠가 재미라는 주관적 요소가 판단 기준이 되기 때문이기도 하지만, 자동화를 위하여 추가적인 개발 부담이 있기 때문이다. 최근 들어, 기존의 NPC 제어 위주로 적용되던 AI 기법에서 벗어나 유전자 알고리즘, 인공 신경망 등의 진화형 연산을 게임 개발에 본격적으로 적용하려는 시도가 지속적으로 이루어지고 있다. 관련된 연구로는 유전자 알고리즘을 사용하여 NPC들의 외형 디자인을 진화시키거나[4], 회피-추격 문제에서 NPC들에게 학습을 시키거나[5], 인공 지능 캐릭터의 성능을 지속적으로 발전시킨 연구가 있다[6]. 또한 Q-Learning 기법을 사용하여 슈팅 게임에서 좀 더 지능적인 움직임을 생성시키거나[7], 신경망 학습을 통해 대전 액션 캐릭터를 진화시키는 연구가 있다[8]. 그 외에 다양한 진화형 알고리즘을 NPC가 아닌 게임 제작 프로세스에 적용하여 게임 제작 자체를 점진적으로 진행시켜 나가거나[3], 레벨 지형을 자동으로 생성하거나[9], 유전자 알고리즘으로 액션 게임의 레벨을 생성하는 연구도 있다[10].

본 논문은 새로운 게임 제작 자동화 기법의 한 방법으로써 기존의 NPC 학습 부분에 주로 적용되었던 유전자 알고리즘을 타워 디펜스 게임의 공격대 생성에 새로이 적용함으로써 레벨 디자이너의 의도에 따라 최적화된 조합으로 공격대를 자동으로 생성하는 것을 목표로 한다.

3. 유전자 알고리즘

유전자 알고리즘은 자연의 진화과정과 유전법칙에 영향을 받은 메타 휴리스틱 알고리즘이다. 유전자 알고리즘은 일반적으로 1)개체의 모집단 정의, 2)적합도(Fitness)에 따른 선택 과정, 3)후손을 생성하기 위한 교배(Crossover), 그리고 새로운 후손 발생을 유도하기 위한 4)돌연변이(Mutation) 단계를 거치며 최적의 해를 탐색한다[11].

유전자 알고리즘을 이용하여 문제에 대한 해를 찾기 위해서는 두 가지 작업이 필요하다. 첫 번째는 풀고자 하는 문제에 대한 가능한 해를 염색체의 형태로 표현하는 것이다. 두 번째는 각 염색체가 문제를 해결하는데 얼마나 적합한지 측정하기 위한 평가함수를 결정하는 것이다. 타워 디펜스 게임에서 공격대 디자인은 구성 요소들의 최적화된 조합을 찾아내는 대표적인 최적화 업무이다. 유전자 알고리즘을 최적화 문제에 적용하기 위해서는 위의 두 가지 작업이 선행되어야 한다. 이를 위해 다음 장에서는 타워 디펜스 게임의 구성요소에 대한 설명을 한다.

4. 타워 디펜스 게임 모델링

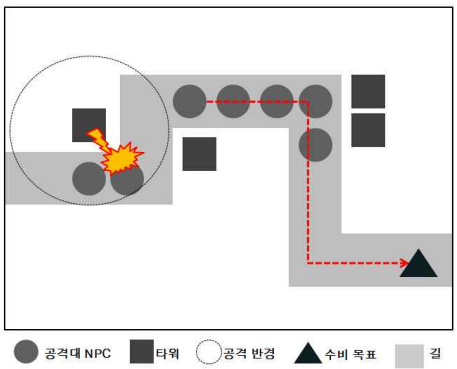
유전자 알고리즘을 적용하기 위해서는 타워 디펜스 게임을 정형화하고 대상이 되는 파라미터를 선별해야 한다. 이 장에서는 다양한 타워 디펜스 게임에서 공통적인 시스템 규칙을 정의하고 이를 기반으로 진화 연산을 적용한다.

4.1 플레이 규칙

본 논문에서는 여러 가지의 타워 디펜스 시스템 규칙 중 본 논문에서는 그 중 가장 범용으로 사용되는 것을 적용하고자 한다. 이는 다양한 타워 디

펜스 게임들을 대상으로 적용을 용이하게 하기 위함이다. [그림 2]는 아래 시스템 규칙을 사용한 타워 디펜스 게임의 모델을 보여준다.

1. 공격대들은 사전에 정해진 길을 따라 이동한다.
2. 하나의 레벨에는 하나 이상의 수비 대상이 존재한다.
3. 공격대는 정해진 시간에 자동적으로 등장한다. 각 레벨 별로 정해진 등장횟수가 존재한다.
4. 공격대가 수비 대상 지점에 도달하면 수비 대상 카운트가 1 줄어들고 해당 공격대도 사라진다.
5. 각 레벨 별로 등장하는 모든 공격대가 지나갔을 때 수비 대상의 카운트가 0이상 남아있을 때 플레이어는 해당 레벨을 클리어하게 된다.
6. 하나의 공격대에는 여러 종류의 NPC가 조합되어 나올 수 있다.
7. 각 공격대 등장 사이에는 지정된 시간이 존재하여 해당 시간 동안 플레이어가 타워를 배치할 수 있는 시간을 준다.
8. 각 타워의 업그레이드는 지원하지 않는다.



[그림 2] 타워 디펜스 게임 모델

4.2 진화 목표값: 난이도

게임의 난이도는 플레이어의 주관적 경험에 크게 의존하는 관계로 이를 정확히 정량화시키기는 현실적으로 어렵다. 그러나 이를 게임 내 사용되는 다양한 수치를 근거로 예상치를 도출해 낼 수는 있다. 본 논문에서 타워 디펜스 게임의 난이도는 수비 목표에 도달한 각 공격대의 공격대 수에 비례한다고 가정한다.

4.3 공격대 파라미터

위와 같은 시스템과 난이도 정의를 기반으로 본 논문에서 각 공격대에서 NPC의 파라미터를 다음과 같이 정의한다.

(1) Health Point (HP)

NPC의 생명력을 나타낸다. HP가 0이 되면 해당 NPC는 사망하게 된다. 높은 HP를 가지고 있을수록 타워의 공격에 오래 살아남을 수 있기 때문에 위협적이다. 높은 HP를 가진 NPC에게는 한번 공격에 많은 데미지를 주는 타워가 유리하고, 낮은 HP를 가진 NPC에게는 데미지는 낮지만 빠르게 공격하는 타워가 유리하다.

(2) 이동 속도 (Speed)

게임 내에서 NPC가 시간당 이동하는 좌표 거리를 나타낸다. 빠른 이동 속도를 가진 NPC일수록 수비에 위협적이다. 이를 저지하기 위해서는 이동 속도를 낮추어 주는 타워를 많이 배치하여야 한다.

(3) 보상 금액 (Reward)

NPC들이 타워에 의해 제거될 때마다 플레이어에게 주는 보상 금액이다. 1, 2번 파라미터가 높을수록 더 높은 보상이 제공되어야 한다.

(4) 등장 간격 (Spacing)

앞에 등장하는 NPC와의 간격을 나타낸다. 등장 간격이 좁을수록 범위 공격을 하는 타워에 쉽게 데미지를 입게 된다.

4.4 타워 파라미터

위와 같은 공격대를 대상으로 본 논문에서는 다음과 같이 타워의 파라미터를 정의한다.

(1) 데미지

타워가 1회 공격할 때 NPC에게 주는 데미지를 나타낸다. 이 수치가 클수록 한번에 NPC HP를 감소시키는 수치가 높아져 공격대 NPC를 짧은 시간에 제거할 수 있게 된다.

(2) 공격 속도

타워가 몇 초 간격으로 공격을 하는지를 나타내는 수치이다. 이 수치가 낮을수록 빠른 공격이 가능하게 된다. 공격 속도가 빠른 타워는 이동속도가 빠른 공격대 NPC에게 효과적이다.

(3) 공격 거리

타워가 NPC를 공격할 수 있는 거리를 나타내는 수치이다. 공격 거리 내에 있는 NPC에게 데미지를 주게 되고, 긴 공격 거리를 가지고 있을수록 공격대에게 위협적인 타워이다.

(4) 범위 공격

한 번의 공격으로 공격대에게 데미지를 입힐 수 있는 범위를 나타내는 수치이다. 클수록 넓은 범위에 광역 데미지를 주게 된다.

(5) 슬루우 공격

NPC가 피격될 때마다 느려지게 만드는 수치이다. 빠른 이동을 하는 공격대 NPC를 느리게 만듦으로써 공격을 용이하게 할 수 있다.

(6) 배치 가격

타워를 배치할 때 소모되는 금액이다. 일반적으로 1, 3, 4, 5번 파라미터 수치가 높을수록, 그리고 2번 파라미터 수치가 낮을수록 배치가격이 증가하게 된다.

타워의 공격대상은 공격 거리 내 가장 먼저 들어오는 공격대 NPC를 공격하도록 하였으며, 해당 NPC가 제거되기 전까지 지속적으로 공격하고, 공격 거리 내에 복수 개의 공격대 NPC가 있을 경우 가장 앞에 있는 공격대 NPC를 공격하도록 하였다.

4.5 진화 알고리즘 대상

본 논문에서는 난이도에 적합한 공격대의 조합 밸런싱을 찾는 것이 목적이다. 이를 위해 진화 알고리즘의 대상은 공격대를 구성하는 NPC의 종류로 설정한다. 타워와 이동 경로는 게임 디자이너에 의해 지정된 데이터를 사용한다.

5. 유전자 알고리즘 적용

본 논문에서는 사전에 게임 디자이너로부터 원하는 수준의 난이도를 입력받고 유전자 알고리즘 기반의 시뮬레이션을 통해 해당 목표치에 근접하도록 반복적으로 공격대의 구성 조합을 시도한다. 유전자 알고리즘을 공격대 구성의 최적화에 적용하기 위하여 다음의 5단계 프로세스를 통해 하나의 세대를 생성하고, 최적화된 값이 나올 때까지 진화시킨다. 각 단계별 구성은 다음과 같다.

5.1 목표치 설정

본 논문에서 목표치는 레벨 디자이너가 설정한 각 레벨의 시간대별 난이도 값이다. 난이도는 해당 공격대가 침입하였을 때 탈취 가능한 수비 대상의 수로 정의한다. 하나의 게임 레벨 내에서 시간대별 난이도 목표치를 설정하면, 시스템은 이를 만족시키기 위해 각 공격대를 생성하게 된다.

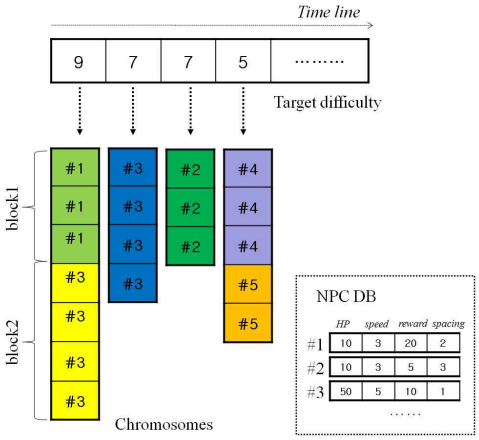
레벨 디자이너로부터의 난이도 입력은 일반적인 테이블 입력을 통해 이루진다. 본 시스템에서는 이러한 시간대별 난이도 입력 방식을 선택함으로써 게임 레벨 내에서 플레이어가 경험할 수 있는 난이도를 직관적으로 배정할 수 있도록 한다.

5.2 유전자 생성

본 논문에서 풀고자 하는 문제는 시간대별 난이

도 값에 수렴하는 최적화된 NPC 구성 조합을 찾는 것이다. 이를 위해 본 시스템에서는 시간대별로 염색체들을 구성한다. 게임 디자이너가 각 시간대별 목표치를 정수형 수치로 입력하면 시스템은 그 수치를 목표로 염색체를 생성한다. 염색체는 동일한 종류의 NPC로 구성되는 블록들의 조합으로 구성되어 있다. 블록들은 교배 시 최소 단위로 이용된다. 블록은 게임 내에서 한 종류의 NPC가 최대 20마리까지 연속으로 나올 수 있다고 가정하여 1~20까지 임의의 길이로 생성된다. 이는 타워 디펜스 공격대가 일반적으로 동일한 종류의 NPC들이 연속적으로 출현하는 것을 표현하기 위함이다.

[그림 3]은 본 논문의 염색체 인코딩 구조도이다. 상단의 배열은 레벨 디자이너가 입력한 시간에 따른 난이도 값이다. 이에 연결된 세로 배열들은 각 난이도를 대상으로 생성된 염색체들을 나타낸다. 각 염색체 내를 구성하고 있는 블록들은 각각 다른 색으로 표시되어 있다. 그림 내 오른쪽 하단은 염색체 생성 시 참조되는 사전에 정의된 NPC 데이터베이스를 나타낸다. 하나의 염색체 안에는 사전에 정의된 다양한 NPC들이 데이터베이스로부터 참조되어 배정된다. 매 세대별 군집체 수는 실험적으로 가장 좋은 결과를 내는 8개를 유지한다.



[그림 3] 염색체 인코딩 구조

5.3 적합도 측정과 염색체 선택

유전자의 적합도를 측정하기 위해 본 논문에서는 시뮬레이션 결과 값을 사용한다. 사전에 지정된 맵과 타워 배치 하에서 생성된 염색체를 기준으로 공격대를 자동으로 생성하였을 때 수비 대상에 도달한 공격대 NPC의 수와 입력된 목표치의 차이를 최소화할 수 있는 염색체를 선택하고자 한다. 생성된 8개의 염색체를 대상으로 시뮬레이션 결과 값과 목표치의 차이를 기준으로 순위 기반 선택 방법 (Ranking Selection)을 사용하여 최상위 2개의 염색체가 선택되도록 한다.

본 시뮬레이션에서는 매 시간별로 선택된 염색체들 보상 총합이 모두 다음 세대 타워 강화에 사용되도록 한다. 즉 시뮬레이션에서 매 시간대별로 타워는 이전 공격대를 물리침으로써 획득된 보상 총액만큼 자동적으로 성장한다. 성장 규칙은 길 주변으로 모서리 부분에 우선적으로 사전에 정의된 타워가 배치하도록 설정한다.

5.4 교배

교배는 2개의 개체 간에 염색체를 부분적으로 서로 바꿈으로써 새로운 개체를 생성하는 단계이다. 이 때 부모의 형질이 자손에게 적절히 계승되는 형질 유전성 (Character Preservingness)을 가질 수 있도록 블록 단위로 균일 교배(Uniform Crossover) 방식을 선택한다. 이것은 각 유전자가 독립적으로 교환될 수 있도록 하는 것으로서 마스크를 씌워 비트가 0일 경우에만 유전자를 교환한다. 마스크 설정의 기준은 각 블록 별로 일정 HP, 이동 속도, 보상 금액, 등장 간격 파라미터들의 총합이 각각 높은 것을 우선 대상으로 선별함으로써 교배 시 파라미터에 좀 더 다양한 변화가 생길 수 있도록 한다.

5.5 변이

변이는 개체에 근접한 새로운 개체를 생성하는 국소적인 랜덤 탐색의 일종이다. 본 논문에서는 돌

연변이 확률을 1.0% 수준으로 설정하여 변이시킨다. 변이로 선택된 블록은 내부 NPC들을 NPC DB로부터 참조하여 재생성한다.

5.6 공격대 생성

유저가 입력한 난이도 값과 가장 유사한 난이도를 생성해 내는 염색체 리스트 조합이 선별되게 되면 시스템은 이 염색체들을 기준으로 공격대를 생성하게 된다. 이렇게 생성된 공격대 리스트는 별도의 사용자 개입 없이 자동으로 생성되어 저장된다. 하나의 레벨을 대상으로 여러 번의 진화 연산을 통해 다수의 공격대 후보가 생성된다.

시스템에서는 이렇게 생성된 다수의 공격대 후보를 다시 우선순위를 두어 정렬한다. 우선순위 기준으로는 데이터 저장 시 레벨 단위로 목표 난이도 값과 시뮬레이션 난이도 값과의 분산 값을 사용한다. 이는 전반적으로 고른 적합도를 가진 공격대를 선별하기 위함이다. 레벨 디자이너는 저장된 공격대 데이터를 우선순위대로 확인하여 최종적으로 해당 레벨에 적합한 공격대를 선택하게 된다.

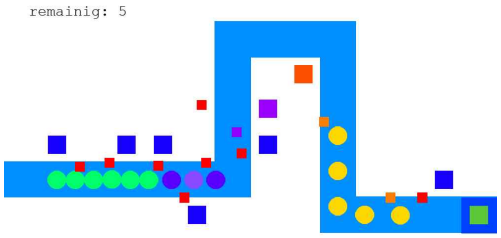
6. 실험 결과

본 시스템은 Windows XP 운영 체제, Intel Core2 Duo 1.83GHz, 1GB 메모리의 환경에서 C++와 DirectX 9.0을 사용하여 구현되었다. 유전자 알고리즘을 제어하기 위한 주요 전역 변수 값들은 최대 세대수는 1,000회, 집단의 크기는 8, 돌연변이 확률은 1.0%로 하였다. 종료 조건으로는 빠른 최적해를 찾는 부분에 주안점을 두어 수렴된 난이도 차이 값이 일정 수치 이하가 되면 종료하도록 하였다. 7종류의 공격대 NPC로 조합되는 공격대 구성에 대해 목표치에 대한 95% 수준의 최적해를 찾는데 소요된 평균 세대수는 약 820 세대였다.

본 논문에서는 4.1절의 규칙을 기반으로 하는 간략한 타워디펜스 게임을 만들어 레벨 내 공격대를 생성하였다. [그림 4]는 본 시스템에 의해 생성된 데이터를 기반으로 한 게임 플레이 화면이다. 해당 레벨에 총 10회의 공격대가 생성되도록 하였고, 사용되는 타워의 리스트는 [표 1]과 같다. 배치되는 타워의 종류는 가장 범용으로 쓰이는 타워로 최대 5종류로 한정하였으며, 각 실험에 사용된 사전에 입력된 공격대 데이터는 [표 2]와 같다.

각 실험은 공격대를 구성하는 NPC의 종류를 변경해 나가면서 유전자 알고리즘의 진화 결과를 생성하였다. 첫 번째 실험은 3종류의 공격대 NPC (S2, N, H1 type)를 대상으로 하였으며, 얻어진 공격대 자동 생성 결과는 [표 3]과 같다. 표에서 괄호 안의 숫자는 생성된 NPC의 수를 의미한다. 실험에 입력된 난이도 값은 후반부로 갈수록 점점 어려워지지만 중간에 약간의 휴식기를 주어 난이도의 강약을 주었다.

두 번째 실험은 5종류의 공격대 NPC(S1, S2, N, H1, H2 type)를 대상으로 [표 4]와 같은 결과를 얻었으며. 세 번째 실험은 7종류의 공격대 NPC (L, S1, S2, N, H1, H2, B type)를 대상으로 하였으며, 얻은 결과는 [표 5]와 같다.



[그림 4] 타워디펜스 게임 플레이화면

본 논문에서 생성된 공격대의 게임 플레이에 대해 실제 유용성을 확인해 보기 위해 HCI에서 주로 사용하는 평가기법 중 하나인 휴리스틱 평가 (Heuristic Evaluation)[12] 방법을 사용하였다. 평가 과정은 다음과 같다. 첫째로 유사한 게임 플레이 경험을 지닌 평가단을 구성하기 위해 타워 디펜스 게임 경험이 있는 플레이어 20명을 모집하였다.

[표 1] 타워 리스트

타워 이름	데미지	공격 속도	공격 거리	범위 공격	슬로우	배치 가격
속사 타워	10	0.5	1	0	0	30
화력 타워	40	1.5	2	0	0	60
슬로우 타워	0	1	1.5	0	1	45
범위 공격 타워	20	0.75	1.5	1	0	100
장거리 타워	30	2	3	0	0	300

[표 2] 공격대 NPC 종류

NPC	HP	이동 속도	보상 금액	등장 간격
행운형 (L)	10	3	20	1
스피드형1(S1)	10	5	5	1
스피드형2(S2)	30	4	10	1
일반형 (N)	100	2	15	1.5
체력형 (H1)	300	1	30	2
체력형2 (H2)	600	1	50	2
보스형 (B)	500	2	100	2

[표 3] 3개의 NPC로 생성된 공격대 구성

등장 순서	목표 난이도	조합된 공격대
1	1	S2(12) N(2)
2	1	S2(18) N(1)
3	1	S2(5) N(1) H1(1)
4	2	S2(19) N(5)
5	3	S2(13) N(5) H1(3)
6	4	S2(10) N(2) H1(7)
7	4	S2(20) H1(7)
8	3	S2(13) N(5) H1(3)
9	5	S2(3) N(5) H1(8)
10	7	S2(10) N(9) H1(10)

[표 4] 5개의 NPC로 생성된 공격대 구성

등장 순서	목표 난이도	조합된 공격대
1	1	S1(13) S2(14) N(1)
2	1	S1(10) S2(10) N(3)
3	1	S1(5) S2(2) N(2) H1(1)
4	2	S1(1) S2(3) N(2) H1(1) H2(1)
5	3	N(2) H1(1) H2(2)
6	4	S1(4) S2(5) H2(4)
7	4	S1(18) S2(19) N(2) H1(1) H2(2)
8	3	S1(12) H2(4)
9	5	S1(2) S2(3) N(3) H1(1) H2(4)
10	7	S1(5) S2(6) N(19) H1(8) H2(1)

[표 5] 7개의 NPC로 생성된 공격대 구성

등장 순서	목표 난이도	조합된 공격대
1	1	S2(5) N(5)
2	1	S1(5) S2(5) N(1) H1(1)
3	1	S1(6) S2(10) N(3)
4	2	S1(19) N(5)
5	3	S1(10) N(3) H1(4) B(1)
6	4	S1(10) N(2) H2(2) B(1) L(4)
7	4	S1(5) S2(19) N(11) H2(1)
8	3	S1(10) S2(5) N(8) H1(5)
9	5	S1(12) S2(13) N(15) H1(1) H2(1)
10	7	S1(5) S2(20) N(12) H1(3) H2(1) B(2)

그리고 이들을 대상으로 본 시스템에 의해 자동으로 생성된 공격대가 포함된 레벨 3개와 게임 디자이너가 실제로 만든 레벨 3개를 섞어서 15분 동안 플레이 한 후 3개의 평가항목에 대해 리커트(Likert) 5단계 척도를 사용하여 평가받았다. 3개의 평가항목과 설문 결과는 [표 6]과 같다.

난이도의 적절성 테스트에서는 평가자의 75%가 보통 이상으로 평가하였고, 자동 생성 인지성 부분에서는 평가자의 70%가 보통 이상으로 평가하였다. 재미 체감도 부분에서는 평가자의 50%가 부정 이하로 평가하였다. 설문 내용을 전체적으로

요약하면 자동으로 생성된 게임 레벨에 대해 대다수가 만족하는 재미는 제공하지 못하였지만, 목표로 하는 난이도의 변화를 느낄 수 있었고, 자동으로 생성된 레벨임을 인지하지 못하였다는 것을 알 수 있었다.

[표 6] 휴리스틱 평가 설문 내용 (단위: %)

평가 항목	설문 내용	매우 부정	부정	보통	긍정	매우 긍정
난이도의 적절성	목표로 한 난이도가 생성되었는가?	10	15	15	25	35
자동 생성 공격대의 인지성	해당 공격대가 시스템에 의해 자동으로 생성된 것을 플레이 도중 알 수 있었는가?	10	20	25	25	20
재미 체감도	타워 디펜스의 재미를 느낄 수 있었는가?	15	35	30	15	5

본 논문에서 제안한 시스템을 통하여 디자이너가 설정한 난이도를 가진 공격대를 자동으로 생성할 수 있음을 휴리스틱 평가를 통해 확인하였다. 본 시스템은 많은 시간이 걸리는 난이도 밸런싱 업무를 사전에 경감시켜 주는데 도움이 되지만, 본 시스템에 의해 생성된 공격대가 적절한 난이도를 갖추고 있더라도 재미를 제공하기 위해서는 추가적으로 재미 부가 작업이 이어져야 함을 알 수 있었다. 이러한 결과는 진화 알고리즘의 평가 함수가 난이도 파라미터만을 대상으로 하고 있기 때문에 발생한 것이다. 그러나 본 시스템에 의해 생성된 공격대를 활용함으로써 추가적인 재미 부가 작업도 완전히 새롭게 공격대를 구성하여야만 하는 기존 업무 방식에 비하여 좀 더 빠르게 이루어질 수 있을 것이다.

7. 결 론

본 논문에서는 유전자 알고리즘을 사용하여 범용적인 게임 레벨 생성을 위한 방법을 새로이 제안하였다. 제안된 시스템은 게임 디자이너가 입력한 난이도 값을 기준으로 게임 내 공격대를 조합하여 최적화된 조합을 생성해 낼 수 있음을 보였다.

기존에는 게임 기획 단계에서 공격대의 조합을 생성하는데 오랜 수작업이 필요했으나, 본 논문에서 제안한 방법은 단순한 난이도 목표치의 입력만으로도 공격대를 자동적으로 생성시킬 수 있음을 보여주었다. 생성된 결과가 실제 레벨 내에서 충분한 재미를 제공하기에는 한계가 있지만, 난이도 밸런싱의 부담을 크게 줄일 수 있다는 점에서 게임 실무에서 유용하게 사용될 수 있을 것이다.

참고문헌

[1] D. Fu, Y. Houlette, S. Henke, “Putting AI in Entertainment: An AI Authoring Tool for Simulation and Games”, IEEE Intelligent and Systems, Vol. 17, No. 4, pp. 81-84, 2002.

[2] D. Johnson and J. Wiles, “Computer Games with Intelligence”, IEEE International Fuzzy Systems Conference, pp. 1355 - 1358, 2001.

[3] J. Togelius and J. Schmidhuber, “An Experiement in Automatic Game Design”, IEEE Symposium on Computational Intelligence and Games, 2008.

[4] 김미숙, 강태원, “게임 캐릭터의 진화하는 디자인”, 한국정보과학회 학술발표대회 논문집, 제30권, 제1호, pp. 410-413, 2003.

[5] 권오광, 박종구, “유전 프로그래밍을 이용한 추격-회피 문제에서의 게임 에이전트 학습”, 정보처리학회논문지B, 제15B권, 제3호, pp. 253-258, 2008.

[6] 이면섭, 조병헌, 정성훈, 성영락, 오하령, “대전 액션 게임에서 유전자 알고리즘을 이용한 지능 캐릭터의 성능평가”, 전자공학회논문지, 제41권, 제4호, pp. 119-127, 2004.

[7] 신용우, “강화학습을 이용한 지능형 게임캐릭터

의 제어", 한국 인터넷 정보학회, 제8권, 제5호, pp. 91-94, 2007.
[8] 조병헌, 정성훈, 성영락, 오하령, "대전 액션 게임을 위한 신경망 지능 캐릭터의 구현", 퍼지 및 지능시스템학회논문지, 제14권, 제4호, pp. 383-389, 2004.
[9] 연제혁, 김성수, 임형준, 이원형, "게임 난이도를 고려한 게임지형 자동 생성 기법에 관한 연구", 한국인터넷정보학회 학술발표대회 논문집, 제5권, 제2호, pp. 477-481, 2004.
[10] 강신진, 신승호, 조성현, "유전자 알고리즘을 사용한 게임 레벨 디자인 기법," 한국컴퓨터그래픽스학회 논문지, 제15권, 제4호, pp.13-21, 2009.
[11] 문병로, 쉽게 배우는 유전 알고리즘, 한빛미디어, 2008.
[12] J. Nielsen, and R. Molich, "Heuristic Evaluation of User Interfaces", Proceeding of ACM CHI, pp. 249-256, 1990.

조 성 현 (Cho, Sung Hyun)

1978년 서울대학교 계산통계학과 이학사
1980년 서울대학교 계산통계학과 이학석사
1995년 UCLA 컴퓨터과학과 이학박사
1996년-현재 홍익대학교 게임학부 교수

관심분야 : 게임 프로그래밍, 게임 그래픽스, 게임 인공지능

강 신 진 (Kang, Shin Jin)

2003 고려대학교 정보통신대학 컴퓨터학과 이학석사
2011 고려대학교 정보통신대학 컴퓨터학과 이학박사
2003-2006 소니 컴퓨터 엔터테인먼트 코리아
(Sony Computer Entertainment Korea)
2006-2008 엔씨소프트(NCSoft)
2008-현재 홍익대학교 게임학부 전임강사

관심분야 : 게임 기획, 컴퓨터 그래픽스, 데이터 마이닝