강화학습을 이용한 게임 테스트 자동화

이석기O, 곽호영*
O*제주대학교 컴퓨터공학과
e-mail:dltjrrl11@gmail.comO, kwak@jejunu.ac.kr*

Game Test Automation with Reinforce Learning

Suk-ki LeeO, Ho-Young Kwak*
O*Dept. of Computer Engineering, Jeju National University

● 요 약 ●

본 논문에서는 강화학습을 통한 게임 테스트 자동화를 제안한다. 게임 테스트의 일부가 게임을 플레이라는 것과 강화학습에 기반을 둔 인공 신경망 모델들이 게임플레이에 많은 성과를 거둔 것에 착안하여 테스트 케이스 자동 생성 및 기계학습을 통한 테스트 자동화를 연구하였다. 테스트 관리자를 두어 게임 요소에 필수적인 테스트 케이스를 데이터 조합으로 생성하고, 테스트 케이스를 수행할 인공지능을 기계학습으로 작성하여 자동화 유지비용을 절감한다. 이 모델을 소형 게임에 시험적으로 적용하였고, 정상 작동을 확인하였다.

키워드: 게임(Game), 테스트 자동화(Test Automation), 강화학습(Reinforce Learning)

I. Introduction

한국콘텐츠진흥원의 자료에 따르면 2016년 한 해전 세계 게임시장 매출이 910억 달러(약109조원)에 이르는 것으로 집계됐다[1]. 시장의 경쟁이 치열하며 개발사들은 경쟁력 강화를 위해 개발시간 단축과 품질 향상을 위해 노력한다.

소프트웨어의 개발에 소요되는 총 비용의 50% 이상과 총 기간의 50% 정도가 개발된 소프트웨어의 테스트 작업에 할당된다[2]. 정도의 차이는 있지만 게임도 테스트는 큰 비중을 차지한다.

테스트 자동화는 크게 두 가지로 분류하는데, 첫째는 테스트 자동화(Automated Test)이며, 둘째는 회귀 테스트 자동화 (Regression Test Automation)이다[3]. 전자는 테스트 케이스를 자동으로 생성하고 수행하는 모든 절차의 자동화 방안이고, 후자는 테스트 케이스는 사람이 직접 작성을 하고 반복인 수행만을 도구에 의존하는 방안이다. 따라서 회귀 테스트 자동화는 소프트웨어 변경 시 테스트 케이스를 유지보수 해야 한다. 모바일 게임이나 PC 온라인 게임은 내용이 빈번히 변경되어 국내 게임사 중에 회귀 테스트 자동화에 회의적인 조직도 상당히 많다[4].

본 논문은 게임 소프트웨어 테스트 자동화에 대한 연구이다. 게임 테스트의 일부가 게임을 플레이하는 것과 강화학습에 기반을 둔 인공 신경망 모델들이 게임 플레이에서 거둔 많은 성과에 착안하여 게임 테스트 중 시스템 테스팅의 기능 테스팅에 있어 사람의 개입을 최소화할 수 있는 방안을 제안한다.

II. Preliminaries

1. 소프트웨어 테스팅

아래 그림은 개발 모델에 테스팅을 접목한 V-모델이다. 테스팅 레벨과 대응되는 개발이 표시되어 있다.

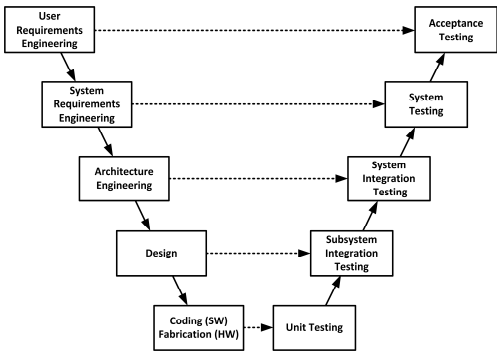

Fig. 1. V-Model[5]

테스팅 대상과 목적에 따라 테스팅 유형이 나뉘는데 기능 테스팅, 비기능 테스팅, 구조적 테스팅, 확인/회귀 테스팅이 있다. 시스템 테스팅 레벨은 소프트웨어의 전체 시스템과 제품의 동작을 테스트한다. 기능 테스팅은 개발된 기능이 요구사항대로 정상 동작하는지 확인하는 테스팅이다. 본 연구의 대상인 시스템 테스팅의 기능 테스팅

은 제품의 전체 기능이 정상 동작하는지 확인하는 테스팅이다. 테스트 과정은 시스템을 블랙박스로 간주하고 테스트 전문 인력이 기능을 확인한다. 즉, 직접 소프트웨어를 실행시켜서 기능을 확인하는 테스트이다.

2. 게임 테스팅

게임에서 시스템 테스팅 단계의 기능 테스팅 중 일부는 게임 플레이다. 이해를 돕기 위해 최근 흥행한 모바일게임 리니지M 표본으로 설명한다.

Fig. 2. Lineage M

Fig. 2를 보면 게임 구성요소로 지형, 나의 캐릭터, 적 등이 있다. 여기서 지형을 테스트 한다면, 지형의 모든 위치에 문제가 없는지 각 지역을 살펴보는 것이다. 그리고 이 테스트를 테스트 담당자는 모든 지형에 수행한다.

3. 국내 게임사 테스트 자동화 사례

NC소프트의 모바일 게임 테스트 자동화 사례[4]는 여러 테스트 단계 중 확인/회귀 테스팅을 자동화한 것으로 테스터가 테스트 케이스와 테스트 스크립트를 작성하고, 자동으로 수행하여 결과를 보고한다.

NHN의 C++언어로 개발된 게임 플랫폼의 테스트 자동화 사례[6]도 확인/회귀 테스팅 자동화 연구이다.

본 연구는 테스트 케이스의 자동생성 및 수행 자동화에 대한 연구로 위 두 연구와 큰 차이가 있다.

4. 강화학습

강화학습은 Fig. 3과 같이 에이전트와 환경으로 구성되며, 에이전트는 행동을 취하고 그 결과로 상태가 변경되고 보상을 획득한다[7].

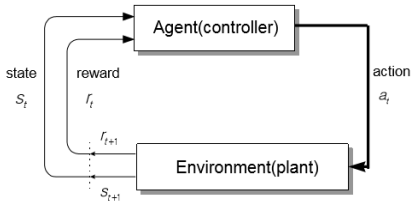

Fig. 3. Reinforce Learning

에이전트는 보상을 이용하여 지속적으로 현재 상태와 행동들의 가치를 측정하며, 행동 선택 정책으로 다음 행동을 선택한다. 상태는 위치 정보 등 소프트웨어 내부데이터를 사용하고, 화면 자료를 인공신경망과 연결하여 시각처럼 이용하는 방법도 있다.

이 외에도 Recurrent Neural Network를 연결하여 기억을 상태정보로 이용하여 성능을 향상시킨 연구도 발표되었다[8].

III. The Proposed Scheme

1. 강화학습을 이용한 게임 테스트 자동화 개요

Fig. 4는 본 논문에서 제안한 테스트 시스템의 처리 방법을 보인 것이다.

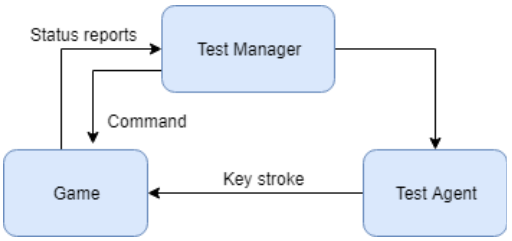

Fig. 4. System Diagram

테스트 관리자가 전체 흐름을 관리하며, 저장된 테스트 케이스 기초 자료에 근거하여 게임의 구성요소를 파악하고, 테스트 케이스를 조합형으로 생산한다.

그리고 각 테스트 케이스와 연결된 보상함수와 행동 선택 정책을 이용하여 강화학습 인공신경망을 학습시킨다. 끝으로, 학습을 완료한 인공신경망을 테스트 에이전트에 적용하여 테스트를 수행하고 결과를 보고한다.

2. 테스트 케이스 자동생성 방안

이해를 돕기 위해 리니지M을 표본으로 설명한다. Table 1은 리니지M의 여러 테스트 중 지역(Map) 검증을 위한 테스트 케이스다.

Table 1. Test case for map level test

Category	Test procedure	Expected response
Map level test	1. Select character	Character can move all movable area
	2. Enter target map	
	3. Check movable area	Character can't move all immovable area
	4. Check immovable area	Character isn't fall in immovable state

이 테스트 케이스를 이동가능 영역과 이동불가능 영역을 각 지역에 따라 변경해 주는 것으로 테스트 케이스를 생성할 수 있다. 그리고 이 값은 지역 데이터의 지형(Terrain) 정보와 배치된 장애물(Obstacle) 정보를 이용하여 계산할 수 있다.

지형 정보를 이용하여 x, y, z축의 최댓값과 최솟값을 확인할 수 있고, 이를 통해 이동가능 영역을 계산한다. 여기에 배치된 사물의 넓이와 높이 자료를 이용하여 이동불가 영역을 계산한다. 그리고

이 계산된 값을 Table 1의 이동가능 영역과 이동 불가능 영역에 대입하는 것으로 테스트 케이스는 자동 생성된다.

3. 테스터 강화학습 인공신경망 학습 방안

앞서 설명한 지역 검증은 이동가능/이동불가능 영역의 유효성을 확인하는 것이다. 이 지역 검증을 수행하기 위해 Table 2에서 보인 세 가지 유형의 테스트 에이전트가 필요하다.

Table 2. Test agent model for map level test

ID	Agent subject	Test object
A1	Explore a specified area	Terrain verification
A2	Try to escape a specified area	Terrain verification
A3	Turn around a specified obstacle	Obstacle verification

Table 3은 각 에이전트를 학습시키기 위한 보상함수와 행동선택 정책이다.

Table 3. Reward function and action policy for map level test agent

ID	Reward function	Action policy
A1	+10point / new position -1point / 3 sec -50point / distance from center	Epsilon greedy approach
A2	+10 point / new position -1point / 3 sec +50point / distance from center	Epsilon greedy approach
A3	+10 point / new position -1point / 3 sec -50point / distance from obstacle	Epsilon greedy approach

지속 시간에 대한 감점은 행동을 지속적으로 최적화하도록 유도하기 위함이다. 새로운 좌표로 이동하였을 때 가산점을 주는 이유는 공통적으로 끊임없이 새로운 좌표로 이동하는 것을 유도한다. 그리고 중심 혹은 사물과의 거리를 배점에 포함하여 대상을 중심에 최대한 접근 혹은 이탈을 유도한다.

탐험적 탐욕 방식은 최선의 행동으로 예측되는 것을 반복하되 일정확률로 임의의 행동을 취하는 것이다. 지역탐색은 계속 새로운 위치를 찾아 이동해야 하므로 탐험적 탐욕 정책이 적절하다.

4. 테스트 결과 판단 및 운영

테스트 결과는 테스트 케이스의 기대 결과와 일치 혹은 불일치 여부이다. 이 확인을 테스트 에이전트가 직접 수행하면 비효율을 야기할 수 있다. 테스트 관리자가 수행하는 것이 합리적이며, 테스트 관리자는 이를 위해 게임과 많은 통신이 필요하다. Table 4는 지형검증을 위해 필요한 데이터 규약이다.

Table 4. Data protocol for map level test

ID	Description	Parameters
1001	Prepare map	Map ID
1002	Prepare character	Character ID
2001	Character position	x-axis coordinate y-axis coordinate

테스트 결과 저장 및 공유는 데이터베이스나 지정 형식 텍스트 파일 등 테스트 환경에 적합한 방법을 선택해야 하며, 결과 확인은 웹페이지 등 여러 방법 중 조직에 적절한 수단을 사용하는 것이 좋다.

IV. Test and Analysis

1. 실험

1.1 환경

실험을 위해 게임, 테스트 관리자, 강화학습 인공신경망을 한 프로그램 안에 구현하였으며 그 내용은 다음과 같다.

- 프로젝트 구성 : 게임엔진 Unity 2017.2.0 버전
- 게임과 테스트 관리자 : 직접 제작
- 인공신경망 시스템 : Unity ML Agents[9] 사용
- 인공신경망 모델 : Proximal Policy Optimization[10]

Fig. 5와 같이 게임은 키보드 방향키로 이동만 가능하다. Table 5와 Table 6은 에이전트의 행동과 상태이다. 에이전트는 항상 이동하고 방향을 조정한다.

Fig. 5. System Diagram

Table 5. Action for test agent

Action index	Description	Data type
1	x-axis direction	float, -1~+1
2	z-axis direction	float, -1~+1

Table 6. State for test agent

St. index	Description	Data type
1	Ratio of valid x-axis coordinates	float, 0~1
2	Ratio of valid z-axis coordinates	float, 0~1
3	Ratio of to check position	float, 0~1

1.2 테스트 케이스

실험 편의를 위해 지형정보는 직접 최대, 최소 영역 값을 파일로

입력하였고, 배치 사물의 이동불가 영역도 위치정보와 이동불가 영역 부피 정보를 직접 입력하였다. 최대, 최소 영역 값을 이용하여 x, z축 2차원 배열을 만들었고, 이동가능 영역과 불가영역을 값으로 입력하였다. 이 2차원 배열이 기대결과로 이동가능 영역이 모두 확인되면 테스트 완수로 처리하였다.

1.3 테스터 인공신경망 학습

보상함수와 행동 선택 정책은 Table 3에서 제시한 것을 적용하였다. 훈련용 지역 3개와 학습 확인용 지역 3개를 준비하였으며, 모두 16x16의 고정크기이고 벽과 장애물을 배치하였다. 학습 결과로 학습 확인용 지역을 수행한 결과 정상적으로 모든 지역을 확인하였다. 사물 배회 에이전트는 모든 지역이 아닌 지정 사물 주변의 이동가능 영역을 모두 확인하는 것으로 완료 조건을 달리하였다.

1.4 테스트 수행 및 결과 수집

테스트 대상은 130x64로 1개를 준비하였다. 배치사물은 10개의 유형으로 총 71개를 배치하였다. 맵의 구역을 16x16으로 나누어 각 구역에 구역 탐색 에이전트와 이탈시도 에이전트를 배치하였고, 각 배치 사물에 사물 배회 에이전트를 배치하였다.

에이전트들은 확인을 위해 일부러 배치한 결함들을 성공적으로 탐색하였다. 테스트 결과는 실험 편의를 위해 발생 순서대로 파일에 좌표를 단순화 하였다.

2. 분석

제시한 기법을 이용하여 성공적으로 테스트가 가능함을 확인하였다. 이 방법을 이용할 경우 여러 에이전트가 나누어 동시에 테스트가 가능하기에 효율이 더 좋다.

복잡한 지형이나 배치 사물의 구조가 복잡한 경우 미확인 지역이 발생할 우려가 있다. 하지만 이는 테스트 담당자가 결과를 확인하여 해당 부분만 직접 확인하는 방법이나, 미확인 지역을 탐색하는 에이전트를 추가하여 보완할 수 있다.

V. Conclusions

기존의 테스트 자동화 방안은 회귀 테스트 자동화에 그쳤고, 게임 콘텐츠의 잦은 변경으로 인해 자동화에 대한 부정적인 시각이 일부 형성되었다. 이에 본 논문은 게임의 시스템 테스팅 중 기능 테스팅에 대한 테스트 케이스 자동 생성 및 기계학습을 통한 테스트 자동화를 제안하였다. 그리고 기초 테스트 중 하나인 지역테스트를 표본으로 자동화가 가능함을 보였다.

향후 게임 테스트의 심도 깊은 분석을 통해 기계학습을 테스트에 적용할 방안을 모색하고자 한다.

REFERENCES

[1] Technical Reports, Korea Creative Content Agency, Vol.1, Jan. 2017.

[2] G. J. Myers, The Art of Software Testing, John Willy & Sons, 1979.

[3] Yury Makedonov, "Manager's Guide to GUI Test Automation," Software Test & Performance Conference, November 3, 2005.

[4] J.W. Kim, "Automate testing of mobile games," Nexon Developer Conference, 2016.

[5] Using V Models for Testing, https://insights.sei.cmu.edu

[6] S.W. Yeum, H.S. Choi, M.S. Jeon, "Test automation example of game platform developed in C++ language," KIPS, 2011.

[7] V. Mnig, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou; D. Wierstra, M. Reidmiller, "Playing Atari with Deep Reinforcement Learning," NIPS, 2013.

[8] G. Lample, D.S. Chaplot, "Playing FPS Games with Deep Reinforcement Learning," Proceedings of 31st AAAI Conference on Artificial Intelligence, 2017.

[9] Unity Machine Learning Agents, https://github.com/Unity-Technologies/ml-agents

[10] Proximal Policy Optimization, https://blog. openai. com/openai-baselines-ppo