

# 좀비 피하기 AI 게임 제작

- 중간보고 -

중간 보고

2020. 05. 12

6조 이 졸업의 끝을 잡고

박지수, 한다란, 남태수, 이유진

# 목차

1. 서론
  - 1.1 연구/개발 배경
  - 1.2 프로젝트 내용
2. 본론
  - 2.1 시스템/GUI 구성
  - 2.2 사용 사례
  - 2.3 적용 기법 및 기술
  - 2.4 핵심 연구/개발 과제
  - 2.5 업무 분담/일정 계획
3. 중간보고
  - 3.1 주요 개발 항목 및 소제목
  - 3.2 주요 개발 항목의 실적
  - 3.3 주요 개발 항목의 향후 계획
4. 결론
  - 4.1 달성 목표
  - 4.2 프로젝트의 의의
  - 4.3 레퍼런스

# 1. 서론

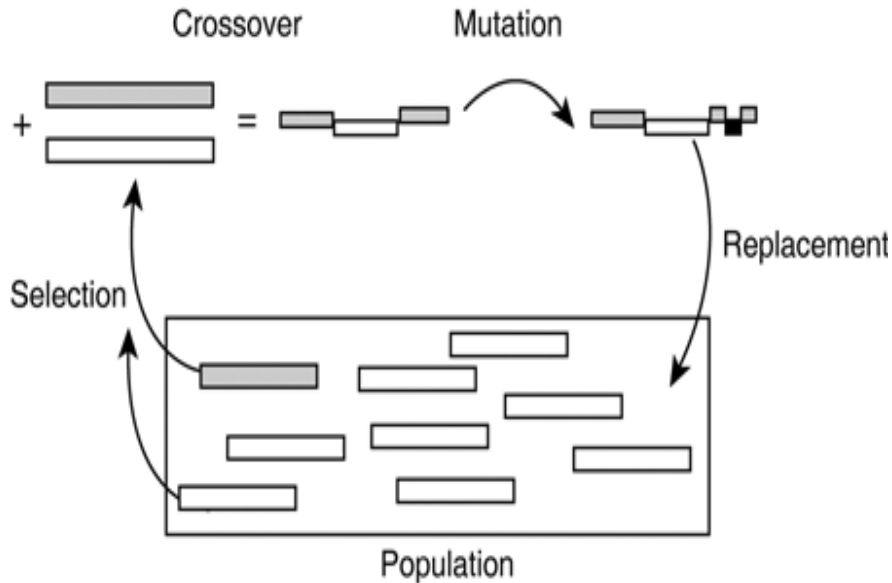
---

1.1 연구/개발 배경

1.2 프로젝트 내용

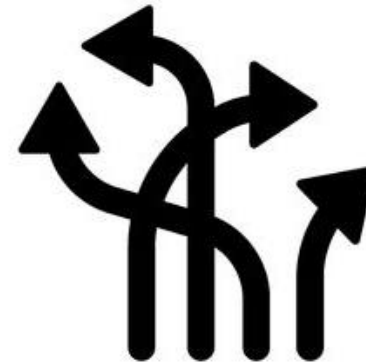
# 1.1 연구/개발 배경

## 1. GA(Genetic Algorithm)의 중요성 부각



- 최근 인공지능과 머신러닝의 급속한 발전으로 GA에 대한 관심 증가
- 수학적으로 명확히 정의되지 않은 문제에도 적용 가능하여 유용성과 확장성을 가짐

## 2. 개체의 유연한 대응 능력의 필요성



**FLEXIBILITY**

- 학습데이터에 대해서는 오차가 감소하는데 비해 실제 데이터에 대해서는 오차가 증가하는 과대적합 (overfitting) 문제 존재
- 다양한 변수의 존재와 증가로 유연한 대응능력의 필요성 증가

# 1.2 프로젝트 내용

## 1단계



## 2단계



## 3단계



## 4단계



## 2. 본론

---

- 2.1 시스템/GUI 구성
- 2.2 적용 기법 및 기술
- 2.3 핵심 연구/개발 과제
- 2.4 업무 분담/일정 계획
- 2.5 개발 환경 및 구현

## 2.1 시스템/GUI 구성

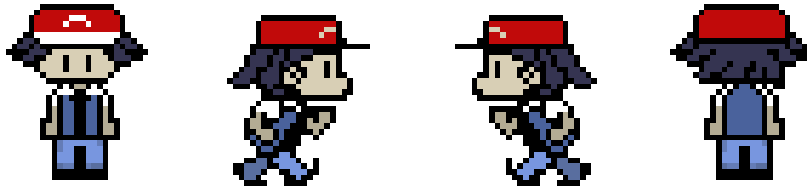
### ● GUI 구성



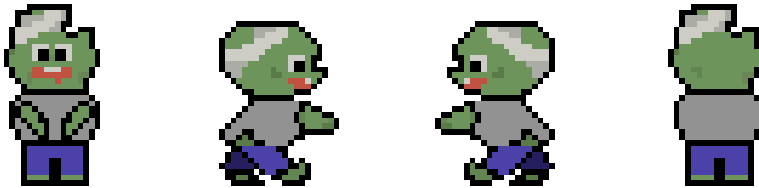
## 2.1 시스템/GUI 구성

- 시스템 구성

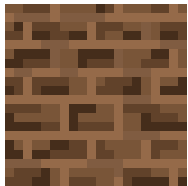
- 플레이어 AI : 좀비를 피해 생존하는 방법을 학습하는 플레이어



- 좀비 : 플레이어AI를 감염시키기 위해 쫓아다니는 게임 빌런



- 벽(맵, 오브젝트) : 게임이 진행되는 배경을 구성함



- 아이템 : 플레이어AI의 지속시간을 연장하는 역할

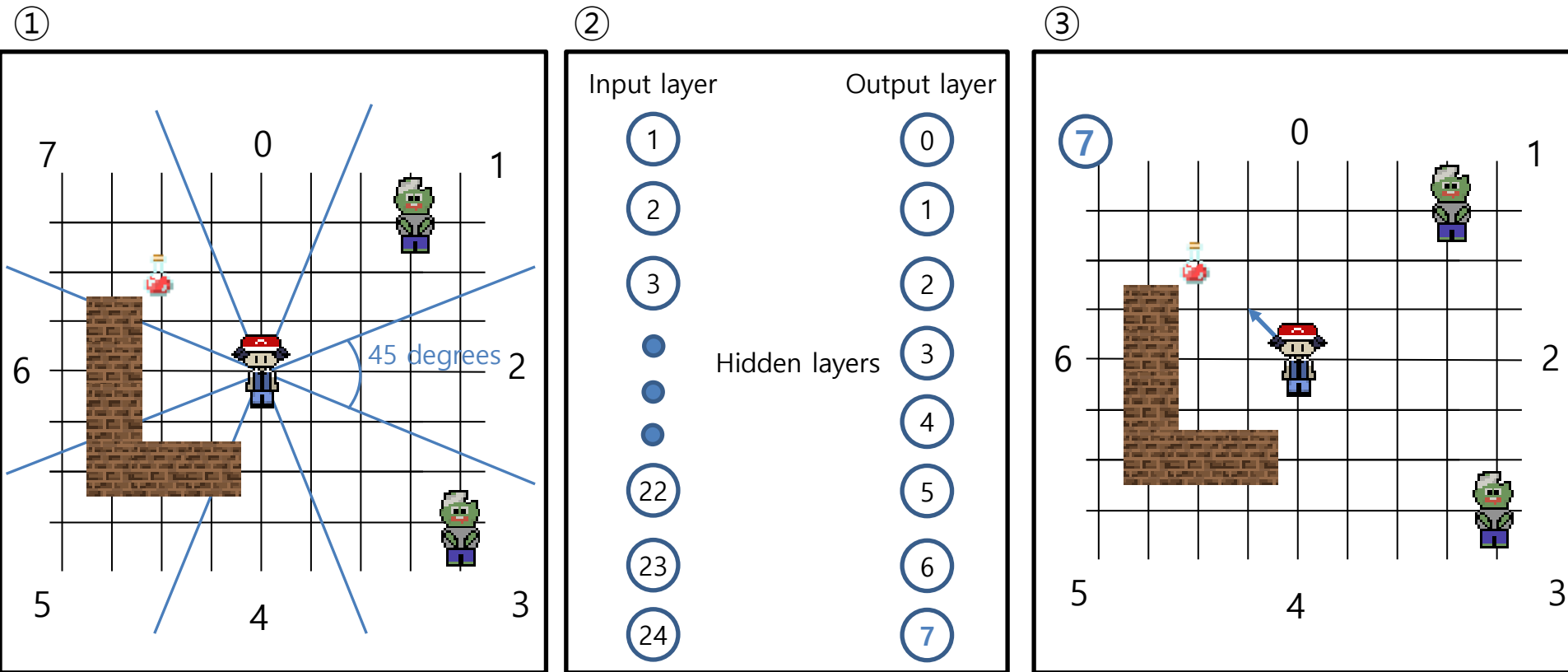




## 2.1 시스템/GUI 구성

■ 플레이어 AI : 좀비를 피해 생존하는 방법을 학습하는 플레이어

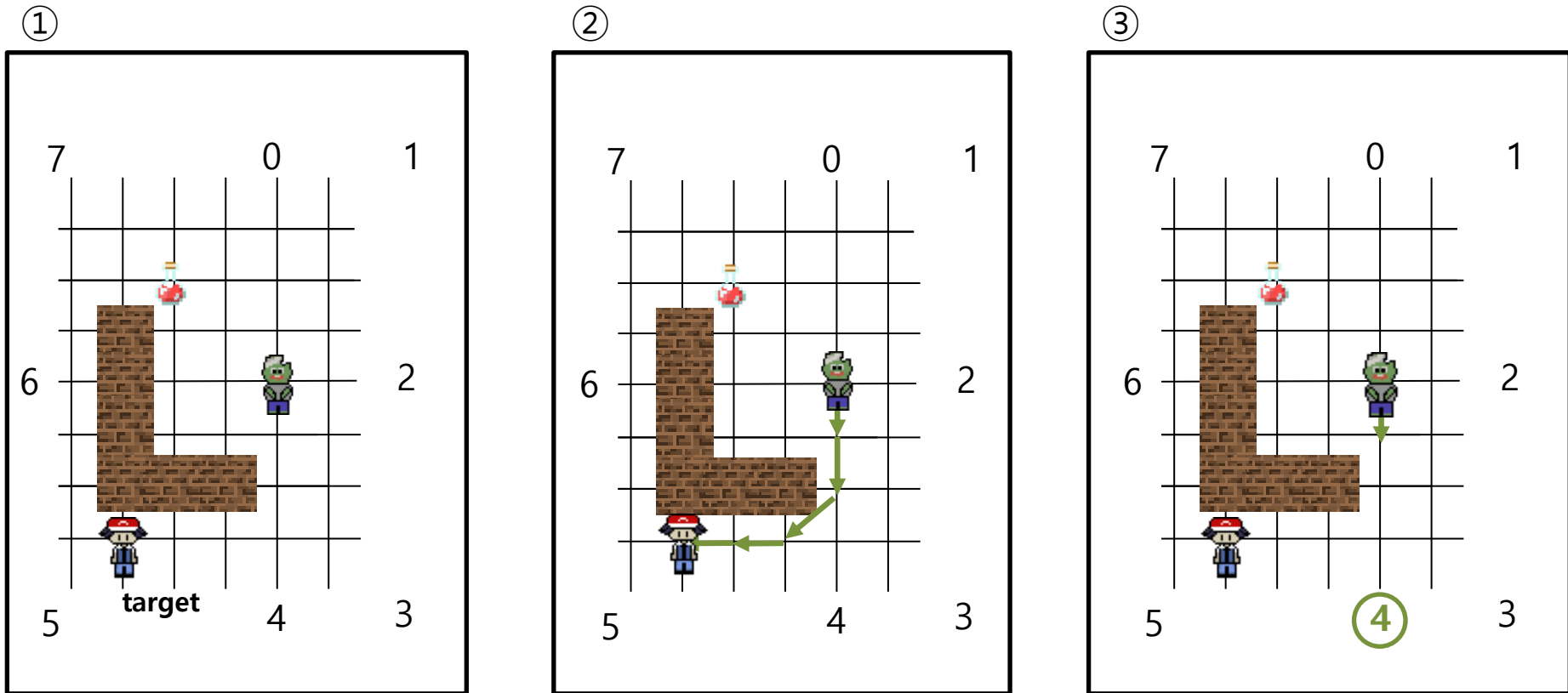
①player-sensor ②neural-net ③step



## 2.1 시스템/GUI 구성

■ 좀비 : 플레이어AI를 감염시키기 위해 쫓아다니는 게임 빌런

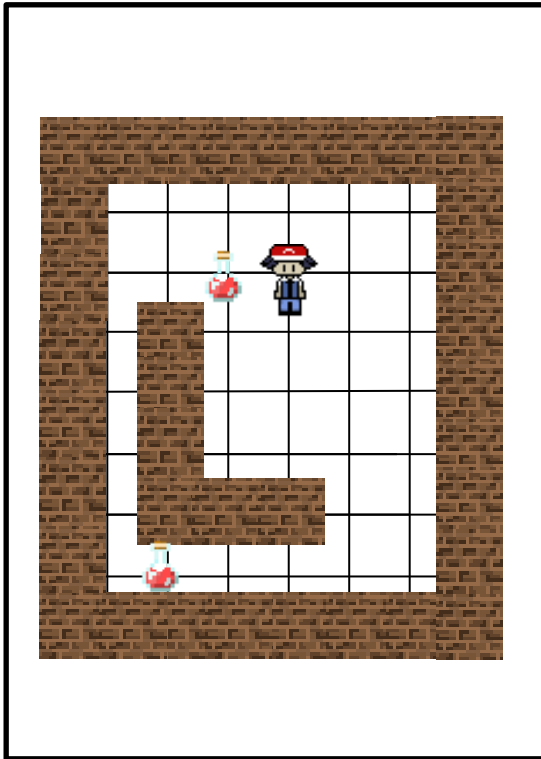
①zombie-sensor ②route ③step



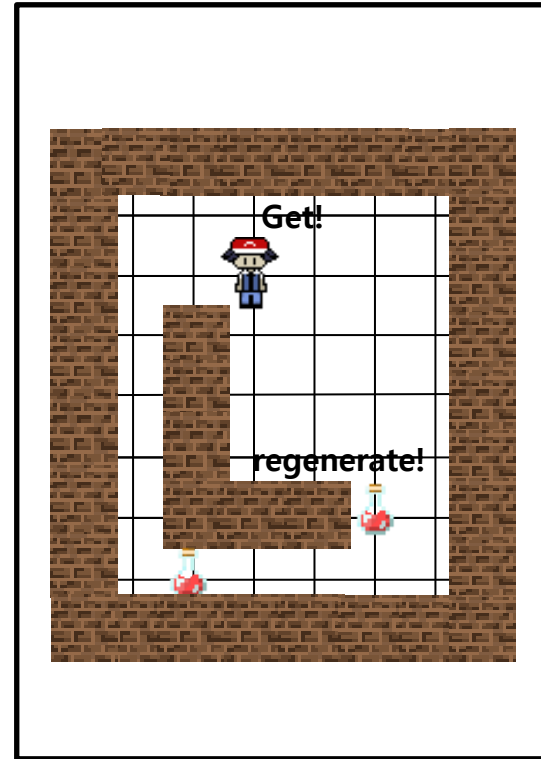
## 2.1 시스템/GUI 구성

- 아이템 : 플레이어AI의 지속시간을 연장하는 역할
- 벽(맵, 오브젝트) : 게임이 진행되는 배경을 구성함

①




②



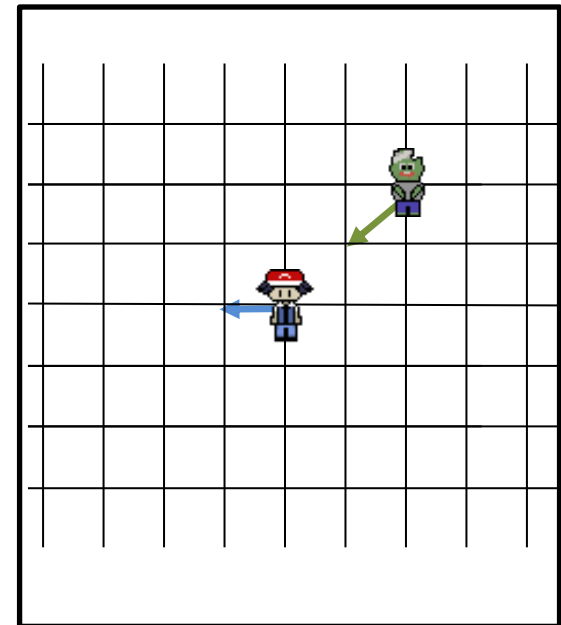
## 2.2 적용 기법 및 기술

- 체비셰프 거리(Chebyshev distance) : 두 지점 사이에서 가장 긴 축 상의 거리를 의미함
  - Distance =  $\text{Max}(|x_a - x_b|, |y_a - y_b|)$
  - 한 번에 이동할 수 있는 거리는 대각선 1방향과 위,아래,좌,우가 같음

distance

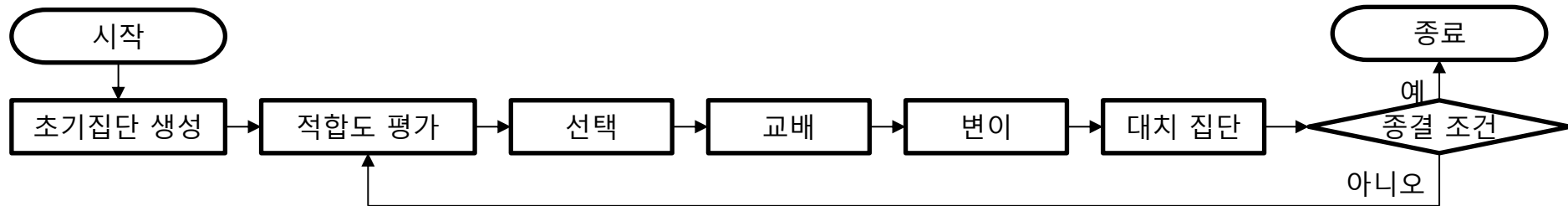
2	2	2	2	2
2	1	1	1	2
2	1		1	2
2	1	1	1	2
2	2	2	2	2

example



## 2.2 적용 기법 및 기술

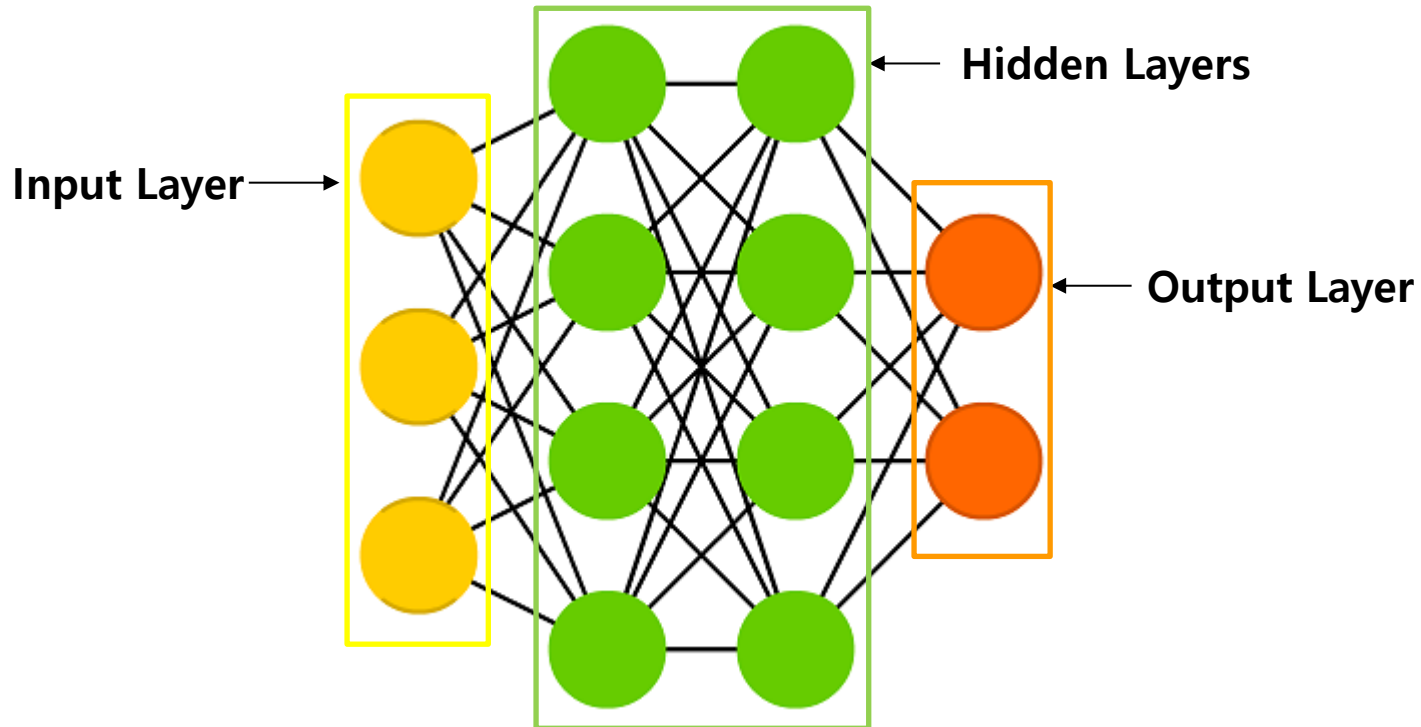
- **유전 알고리즘** : 진화의 핵심 원리인 자연 선택과 유전자의 개념을 이용한 최적화 기법



- **선택(selection)** : 적합도에 따라 다음세대에 유전자를 물려줄 후보 개체를 선별하는 연산
- **교배(Crossover)** : 부모 양측으로부터 받은 유전자를 혼합하여 새로운 개체를 생성하는 연산
- **변이(Mutation)** : 유전자에 직접적으로 변이를 일으켜 해를 변경하는 연산

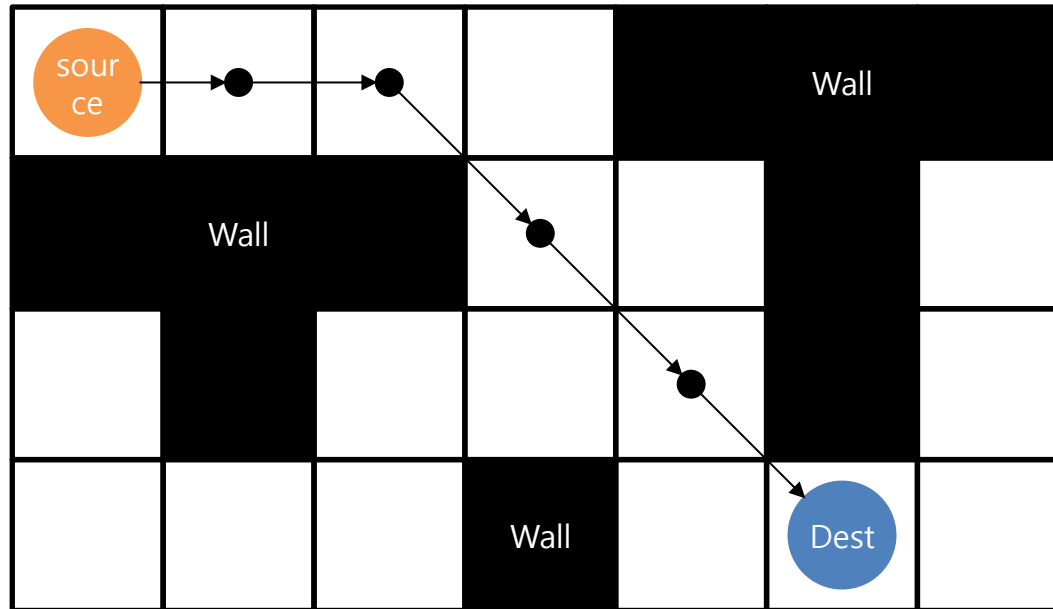
## 2.2 적용 기법 및 기술

- **인공 신경망** : 인간의 뇌가 패턴을 인식하는 방식을 모방한 알고리즘
- **Deep Feedforward Network (DFN)**
  - DFN은 딥 러닝에서 가장 기본적으로 이용되는 인공신경망
  - DFN은 입력층, 은닉층, 출력층으로 이루어져 있으며, 보통은 2개 이상의 은닉층을 이용한다. DFN에서 입력 데이터는 입력층, 은닉층, 출력층의 순서로 전파된다.



## 2.2 적용 기법 및 기술

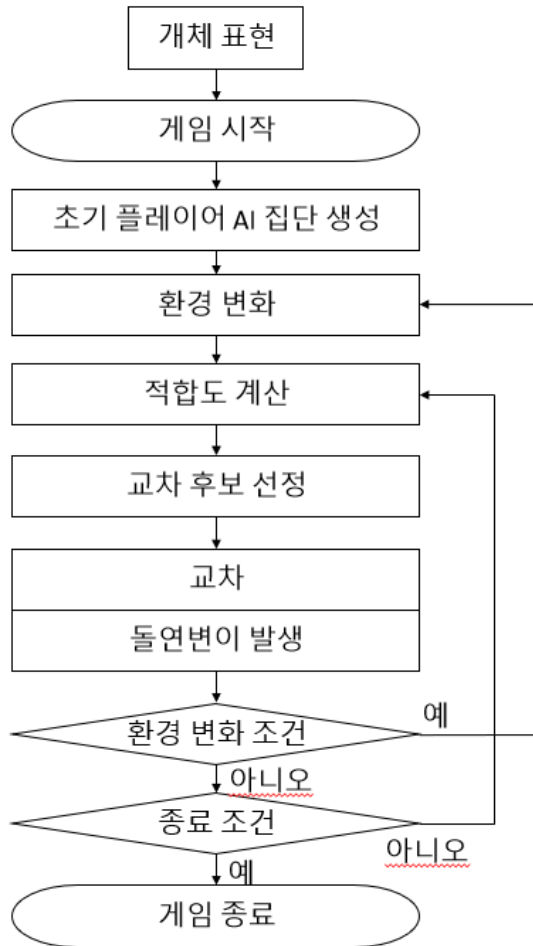
- **A\* algorithm** : 두 지점간 최단거리 파악에 활용하는 알고리즘



- 장애물이 있는 상태에서 목표지점까지의 cost가 제일 적은 경로를 탐색하는 알고리즘
- 좀비가 플레이어를 발견하고 쫓아가는 상황에 적용

## 2.3 핵심 연구/개발 과제

### ● 1. 플레이어AI의 유전 알고리즘 개발



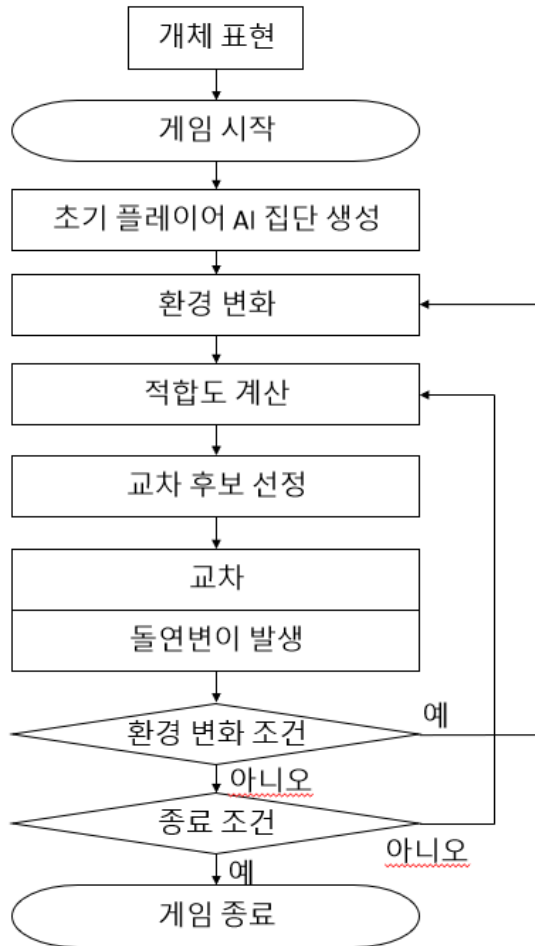
1. 개체표현 : 플레이어AI의 염색체를 코드화하고 적합도함수를 설정
2. 초기AI집단 생성 : 임의의 값을 갖는 염색체를 n개 생성 (1세대)
3. 환경 변화 : 맵의 난이도를 상향
4. 적합도 계산 : 각 플레이어AI의 적합도 계산
5. 선택 : 적합도를 기준으로 룰렛 휠 방식 적용
6. 교차 : 선택된 부모 개체 유전자를 일부 교환하여 자식 세대 생성
7. 돌연변이 : 변이율에 따라 염색체의 일정 부분을 대치하여 지역최적화(local optimization)방지
8. 환경 변화 조건 : 일정 세대에 도달하면 맵의 난이도를 상향
9. 종료조건 : 지정한 세대수에 도달하거나 일정 적합도를 넘어서면 종료



## 2.3 핵심 연구/개발 과제

### ● 1. 플레이어AI의 유전 알고리즘 개발

(1) 물체를 인식하는 시각 센서 구현과 염색체 코드화



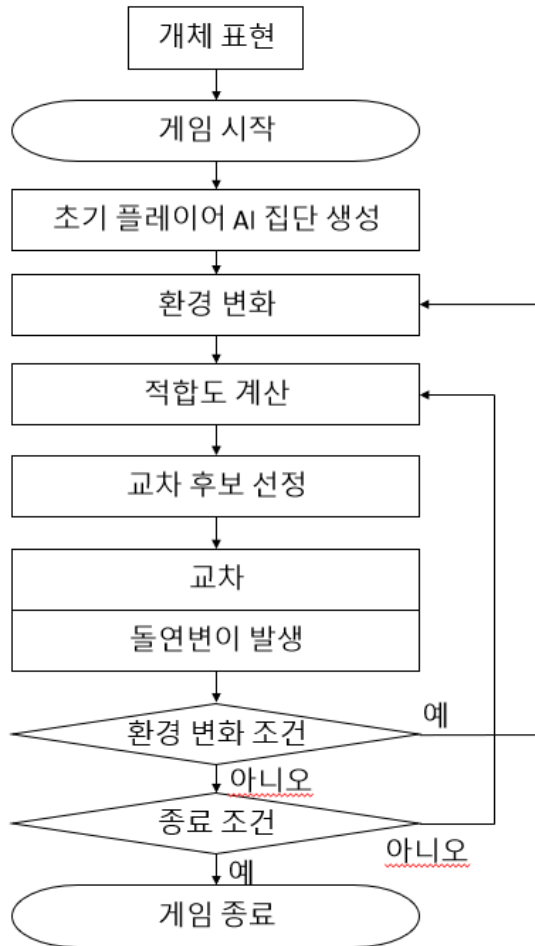
■ 플레이어AI의 염색체를 일 차원 배열로 코드화 : 8방향 센서 \* 3개 타겟(좀비,아이템,벽)으로, 총 24개의 0과 1사이 실수 값을 갖는 배열에 임의의 값을 갖는 염색체를 n개 생성 (1세대)

- ◆ Input value : 1 / 플레이어AI와 센서가 인식한 물체와의 거리  
$$= 1.0 / \text{distance}$$

## 2.3 핵심 연구/개발 과제

### ● 1. 플레이어AI의 유전 알고리즘 개발

#### (2) 유전관련 변수 설정



#### ■ 사전에 결정된 변수

1. 교차 타입: Roulette wheel selection

#### ■ 실험을 통해 조정할 변수

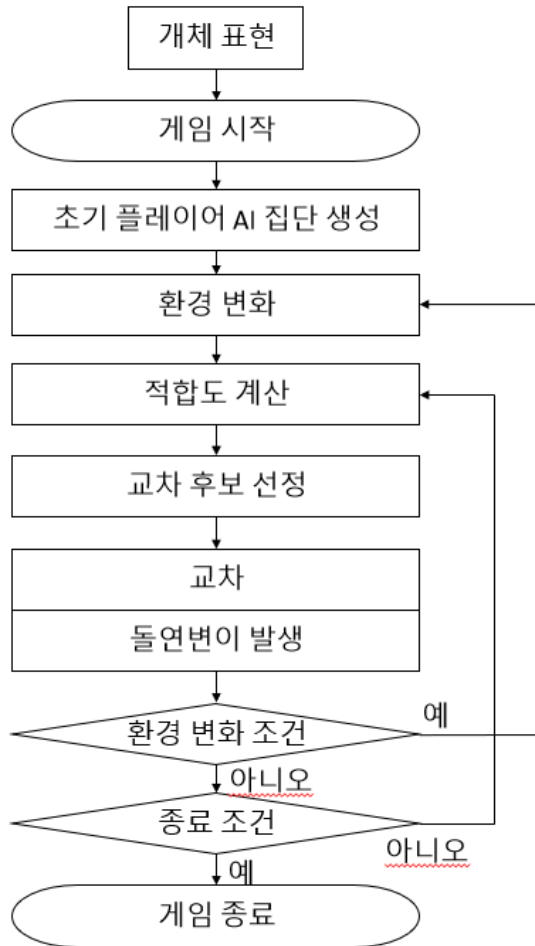
1. 부모 개체 수:  $n$ , 자식 개체 수 :  $m$
2. 교차 타입 : Uniform binary crossover
3. 돌연변이율 : 0.1
4. 돌연변이율 타입 : 'static'
5. 돌연변이 발생 분포 방식 :

Probability random uniform

## 2.3 핵심 연구/개발 과제

### ● 1. 플레이어AI의 유전 알고리즘 개발

#### (3) 적합도 함수 설정



- 개체 적합도를 잘 반영할 함수를 만들어야 함
- 플레이어AI가 의도대로 유전 과정을 거치는 경우

1. 아이템을 향해 접근함
2. 좀비를 만나면 도망 감
3. 벽에 끼어 도망가지 못하는 경우를 기피함

위와 같은 세 가지 특성을 가지며 생존 시간 극대화라는 목적을 달성할 수 있어야 함

$$\text{ex) } F(\text{survival time, item}) = \alpha * \text{survival time} + \beta * \text{item}$$

## 2.3 핵심 연구/개발 과제

### ● 2. 인공지능망 개발

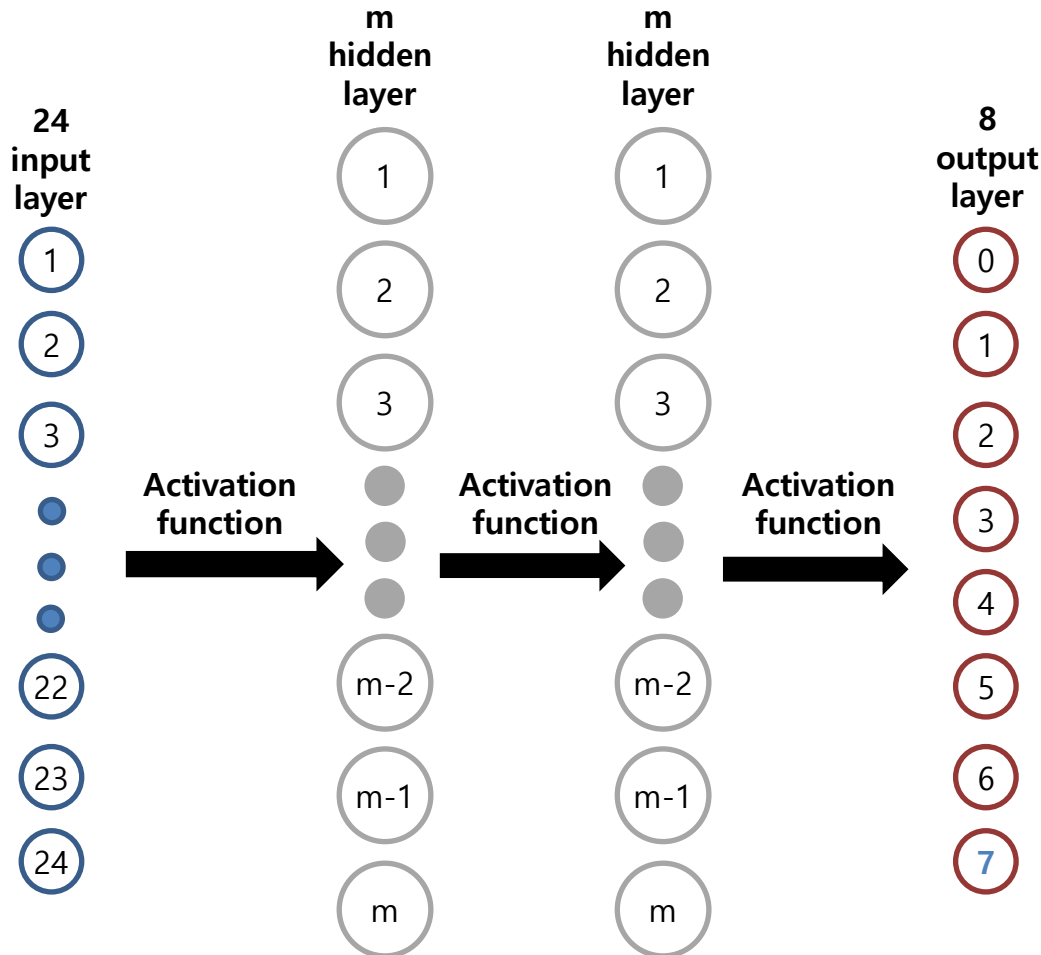
#### (1) 뉴런과 각 layer들의 구성

- Input layer : 플레이어AI의 sensor (24 input neurons)  
-아이템, 좀비, 벽과의 거리를 입력 받음
- Hidden layers : m개의 뉴런을 가진 n층  
-입력 값에 가중치와 오차를 연산하여 다음 층으로 보내는 기저함수의 역할을 함
- Output layer : 8방향 중 최종 출력을 나타낼 수 있는 8개의 뉴런으로 구성

## 2.3 핵심 연구/개발 과제

### ● 2. 인공지능 개발

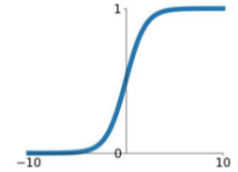
#### (2)인공신경망 구조 및 활성화 함수



## Activation Functions

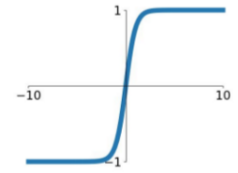
### Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



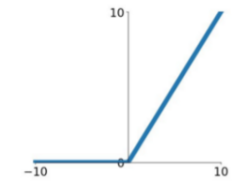
### tanh

$$\tanh(x)$$



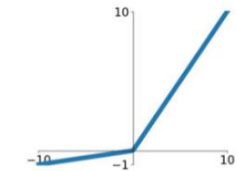
### ReLU

$$\max(0, x)$$



### Leaky ReLU

$$\max(0.1x, x)$$

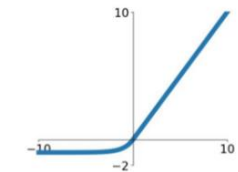


### Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

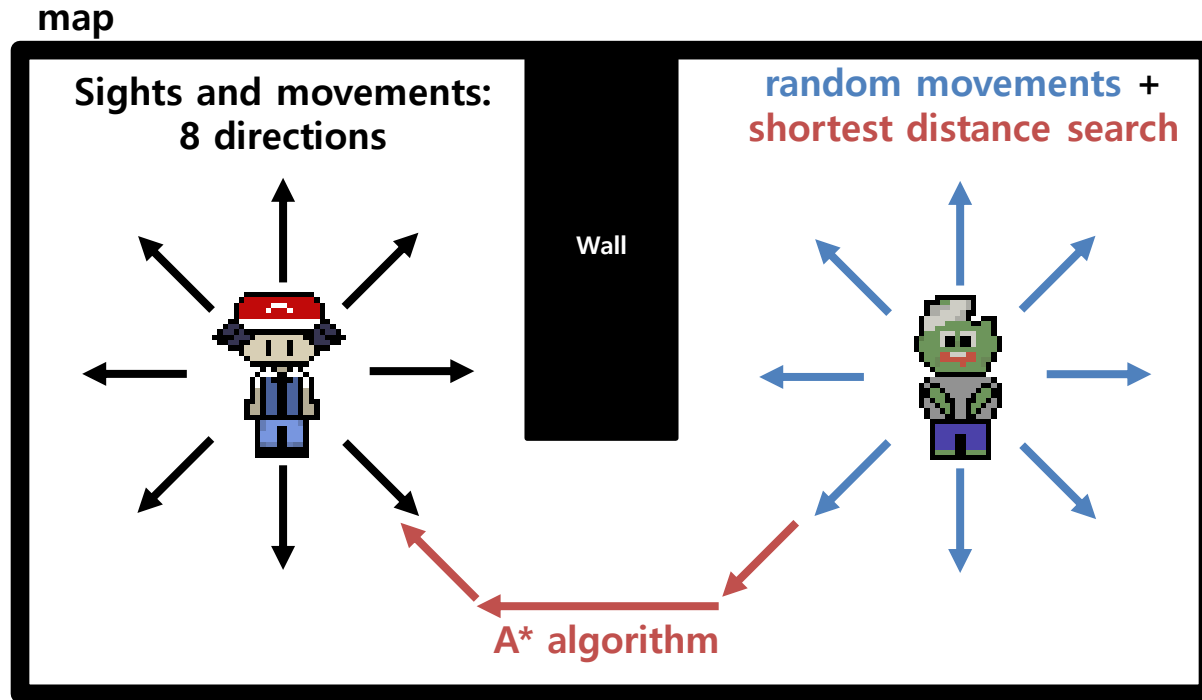
### ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



## 2.3 핵심 연구/개발 과제

### ● 3. 좀비의 행동 알고리즘 개발



- 좀비 또한 플레이어AI와 같이 8방향의 시각센서를 보유함
- 시야 범위 내에 플레이어AI가 포착되었을 경우 A\* 알고리즘을 사용해 최단경로 추적
- 플레이어AI의 위치가 실시간으로 변하는 것을 반영

## 2.4 업무 분담/일정 계획

- **업무분담**

- **이유진**: 게임 GUI 개발
- **남태수, 박지수**: 플레이어 AI 유전 알고리즘, 인공신경망 구현
- **한다란**: 좀비의 최단거리 알고리즘 구현, 벽 및 아이템 속성 알고리즘 구현

## 2.4 업무 분담/일정 계획

### ● 일정 계획



다같이



남태수



한다란



박지수



이유진

	3월				4월					5월					6월		
	Week 2	Week 3	Week 4	Week 5	Week 1	Week 2	Week 3	Week 4	Week 5	Week 1	Week 2	Week 3	Week 4	Week 5	Week 1	Week 2	Week 3
1차 제안서 작성	다같이																
2차 제안서 작성		다같이															
최종 제안서 작성 및 발표			다같이														
게임 환경 설계 및 GUI 디자인					이유진												
플레이어 AI의 유전 알고리즘 설계 및 구현					박지수												
					남태수												
좀비의 최단거리 알고리즘 구현					한다란												
벽과 아이템 속성 알고리즘 구현					한다란												
중간발표: 설계 결과 보고(보고서 및 발표)												다같이					
유전 알고리즘 최적의 적합도 찾기													박지수				
													남태수				
AI 생존시간 비교													박지수				
													남태수				
A* 알고리즘 개선													이유진				
													한다란				
테스트 및 디버깅																다같이	
최종 발표																	다같이



## 2.5 개발 환경 및 구현



### pygame

- 파이썬의 파이게임 패키지를 기반으로 게임 제작
- GUI구현 시 활용



### Git 및 Github

- 원활한 코드 웨어 및 일관성 유지를 위해 사용



### Anaconda

- 플레이어AI 및 전반적인 개발 환경

# 3. 중간 보고

---

3.1 주요 개발 항목 및 소제목

3.2 주요 개발 항목의 실적

3.3 주요 개발 항목의 향후 계획

# 3.1 주요 개발 항목 및 소제목

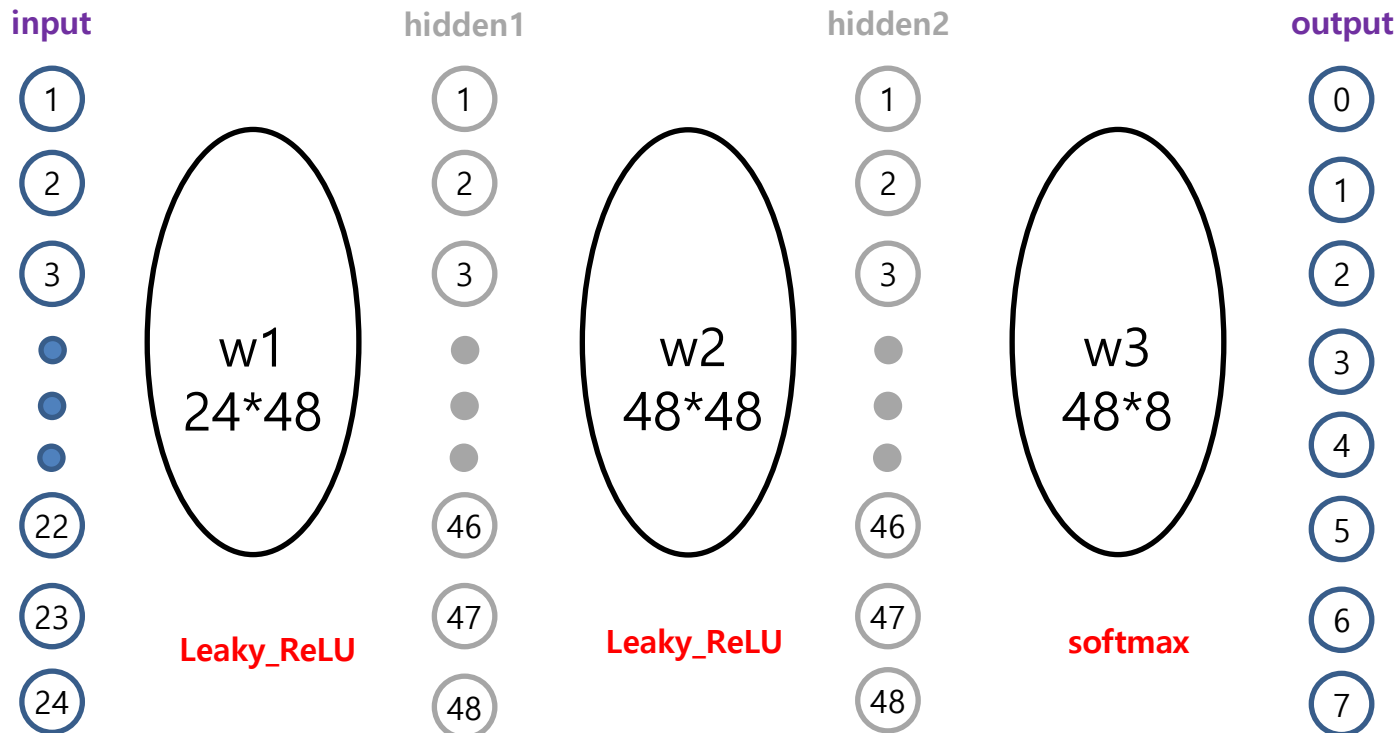
1. 세대를 거칠수록 플레이어AI의 생존시간이 길어지는 유전 알고리즘 구현(75%)
  1. 인공신경망(구현완료)
  2. 유전 알고리즘(최적화 작업중)
2. 단계별로 학습한 AI의 생존시간이 그렇지 않은 AI 생존시간의 130% 이상 달성(50%)
  1. 총 4단계의 맵 구성(구현완료)
  2. 단계적으로 학습한 플레이어AI와 한가지 맵(1단계)에서 학습한 플레이어AI의 생존시간 비교(구현중)
3. 좀비가 장애물을 인지하며 플레이어AI를 쫓는 최단거리 탐색 알고리즘 구현(80%)
  1. A\* 알고리즘을 사용한 최단경로 추적(성능 개선 작업중)
  2. 벽을 인식하여 플레이어를 추적(구현완료)
4. 게임 맵, 아이템, 좀비와 플레이어 AI를 그래픽으로 구현(100%)
  1. 실제 게임과 유사한 GUI 구성(구현완료)
  2. 정적 요소 구현(벽, 아이템)(구현완료)
  3. 동적 요소 구현(플레이어, 좀비)(구현완료)

## 3.2 주요 개발 항목의 실적

### ● 1-1. 인공신경망 개발

(1) 뉴런과 각 layer들의 구성

- Hidden layer : 입력 값에 가중치와 오차를 연산하여 다음 층으로 보내는 기저함수의 역할



## 3.2 주요 개발 항목의 실적

### ● 1-1. 인공신경망 개발

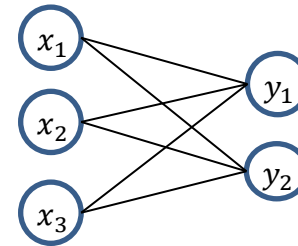
(1) 뉴런과 각 layer들의 구성

#### ■ 가중치 설계

- 가중치는 GA 에서 염색체 역할을 하며, 입력된 신호가 결과에 주는 영향력을 조절하는 요소로 작용함

$$\text{ex) } y_1 = x_1 w_{11} + x_2 w_{21} + x_3 w_{31}$$

$$\text{ex) } y_2 = x_1 w_{12} + x_2 w_{22} + x_3 w_{32}$$



#### ■ 가중치 초기화

- 표준정규분포를 따르는 난수배열이며, ReLU계열 활성화 함수의 초기화 방식인 He 초기화를 하여 구현

He initialization : 가중치를 인풋 개수의 절반의 제곱근으로 나누어 줌

## 3.2 주요 개발 항목의 실적

### ● 1-1. 인공신경망 개발

(2) 인공신경망 아키텍처 설계

■ Hidden layer activation function : 'leaky\_ReLU'

■ Output layer activation function : 'softmax'

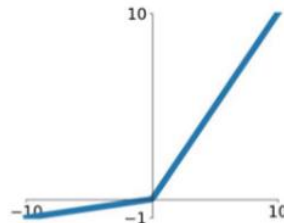
- 함수의 출력을 '확률' 로 해석 가능

출력범위 : 0 ~ 1.0      출력의 총 합 : 1

■ Hidden network architecture : [48, 48]

Activation Functions

**Leaky ReLU**  
 $\max(0.1x, x)$



softmax

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^n \exp(a_i)}$$

## 3.2 주요 개발 항목의 실적

### ● 1-2. 플레이어AI의 유전 알고리즘 개발

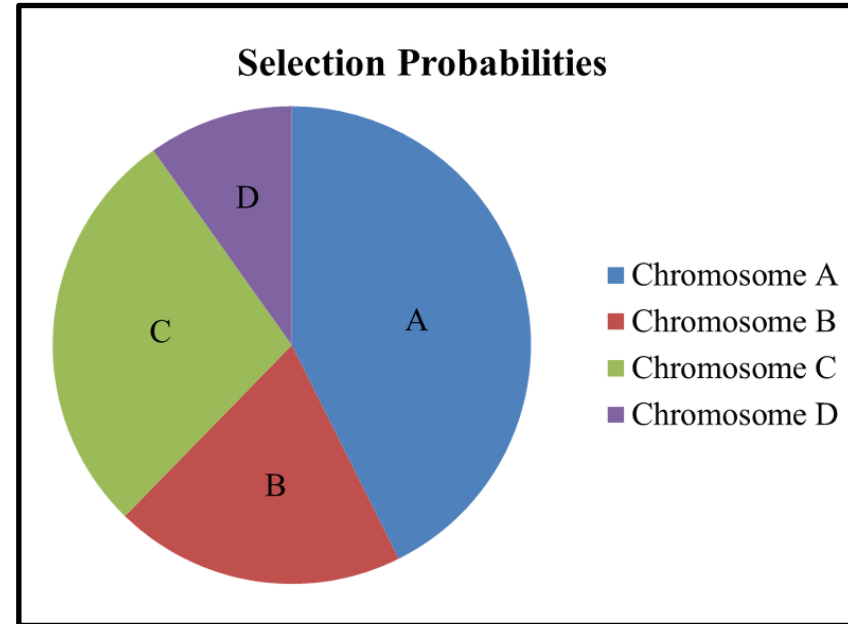
: 유전관련 변수 설정

1. 선택 방식 : Roulette wheel selection

- 점수(적합도)가 높은 개체일수록 점수에 비례하여 선택될 확률을 크게 설정

- 중복되지 않게 설정된 부모 수 만큼 선택 (부모: 10, 자식: 10, 테스트 버전)

- 다만 점수의 차가 너무 많이 벌어지면 하위권 개체들은 선택의 기회가 거의 없어지므로 점수를 스케일링 하여 어느정도 확률을 보장 (비율 : 4)

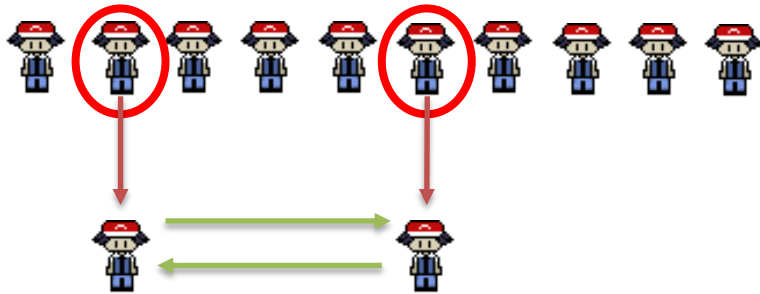


## 3.2 주요 개발 항목의 실적

### ● 1-2. 플레이어AI의 유전 알고리즘 개발

: 유전관련 변수 설정

2. 교차 타입 : Uniform binary crossover



Mask array


- 뽑힌 부모군에서 무작위로 2명을 선택
- 부모 염색체를 그대로 복제한 자식 2명 생성
- Mask 배열을 생성해 평균값인 0.5를 넘으면 마스크 위치 배열의 원소를 자식간에 서로 교체
- 자식 수가 지정된 수가 될 때까지 반복



## 3.2 주요 개발 항목의 실적

### ● 1-2. 플레이어AI의 유전 알고리즘 개발

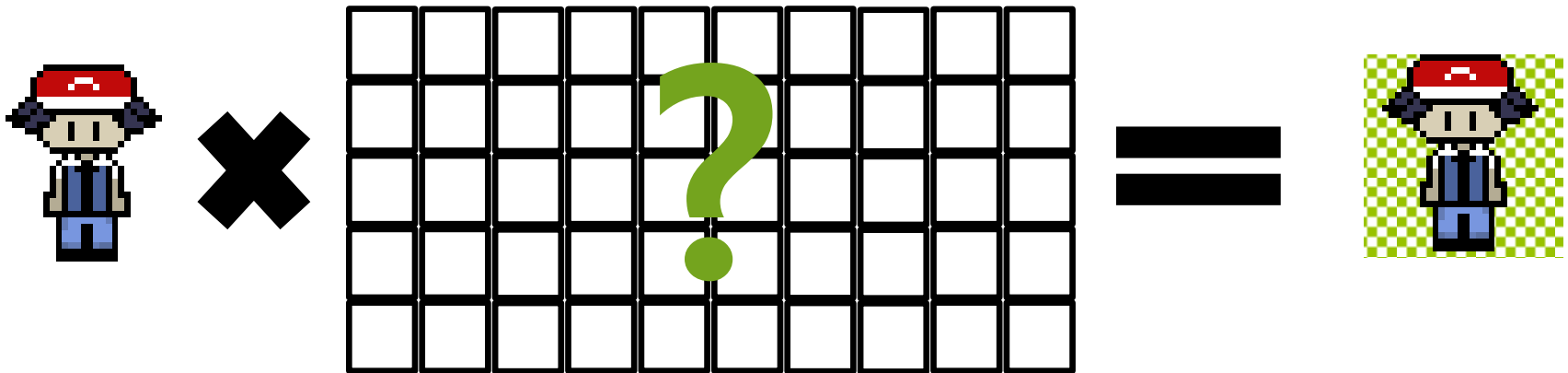
: 유전관련 변수 설정

3. 돌연변이율 : 0.1 (테스트 버전)

4. 돌연변이율 타입 : 'static' , 세대를 거듭해도 변이율이 줄지 않음

5. 돌연변이 발생 분포 방식 : Probability random uniform(-2, 2)

- 교차 과정을 거쳐 생성된 자식과 뽑혔던 부모 염색체 배열에 확률적으로 균등분포 난수를 곱하여 돌연변이 염색체를 생성

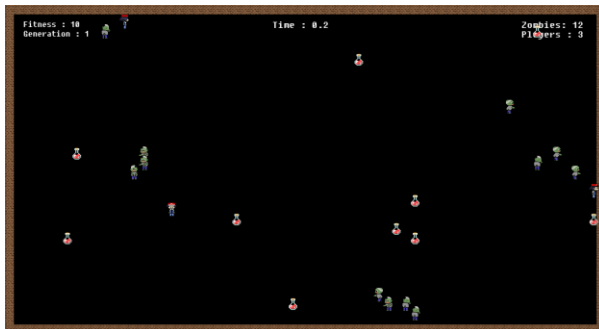


## 3.2 주요 개발 항목의 실적

- 2. 단계별로 학습한 AI의 생존시간이 그렇지 않은 AI 생존시간의 130% 이상 달성

(1) 총 4단계의 맵 구성

- numpy 라이브러리를 이용해 벽 좌표 생성

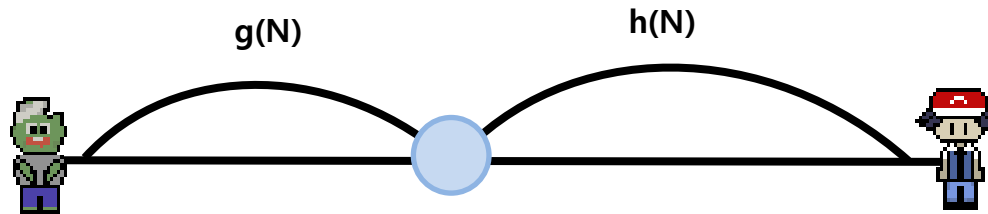


## 3.2 주요 개발 항목의 실적

- 3. 좀비가 장애물을 인지하며 플레이어AI를 쫓는 최단거리 탐색 알고리즘 구현

- **A\* algorithm** : 두 지점간 최단거리 파악을 위해 이용

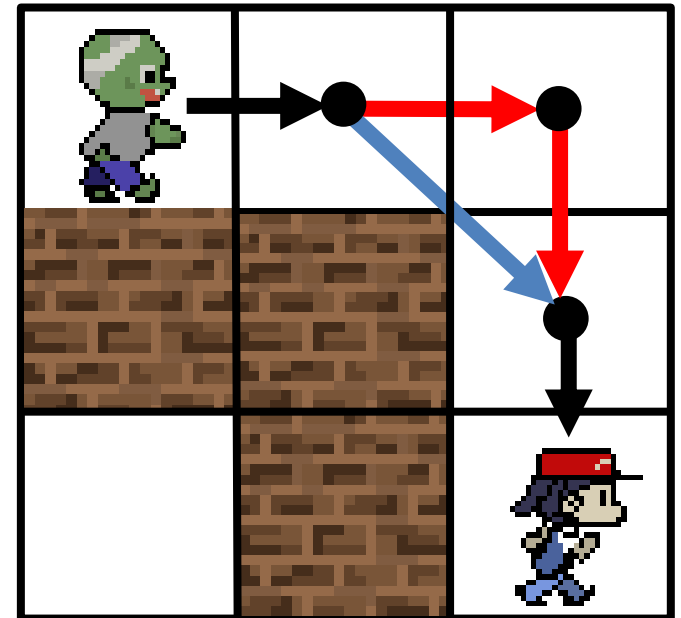
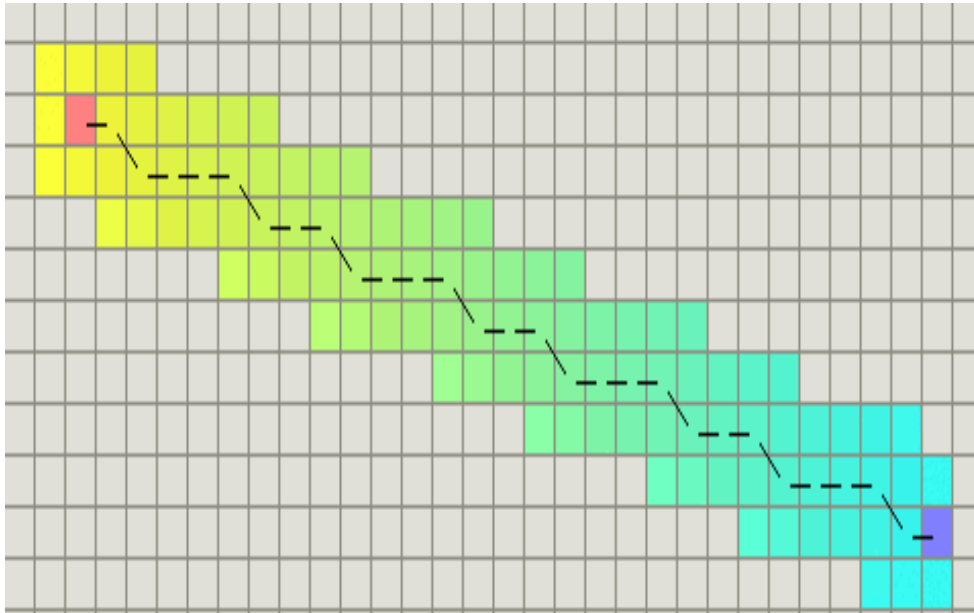
$$f(N) = g(N) + h(N)$$



- **g(N)** : 초기의 좀비 위치에서 현재 위치까지의 최단거리
- **h(N)** : 현재 위치에서 플레이어AI까지의 최단거리 추정치
- **f(N)** : g(N)과 h(N)을 합한 최단거리

## 3.2 주요 개발 항목의 실적

- 3. 좀비가 장애물을 인지하며 플레이어AI를 쫓는 최단거리 탐색 알고리즘 구현
  - **Chebyshev distance** : 직선방향과 대각선방향을 구분하지 않고 이동하여 최소스텝으로 목표물까지 이동가능



## 3.2 주요 개발 항목의 실적

### ● 4. 게임 맵, 아이템, 좀비와 플레이어 AI를 그래픽으로 구현

#### 1. 게임과 유사한 GUI 구성

- 게임 개발에 pygame 라이브러리 사용
- 사용한 폰트: DungGeunMo 폰트(상업적 무료 폰트)  
(<https://cactus.tistory.com/193>)
- Sprite 디자인 툴: Piskel(<https://www.piskelapp.com/>)

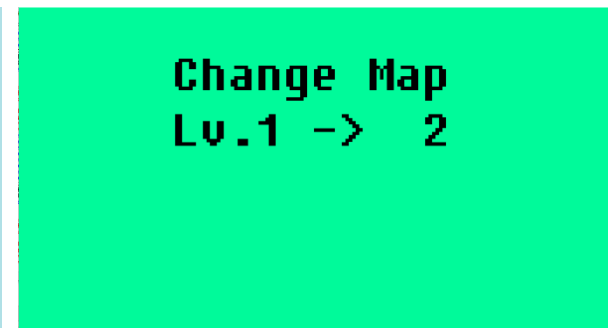
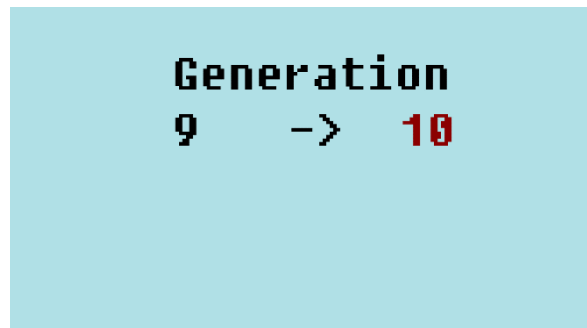
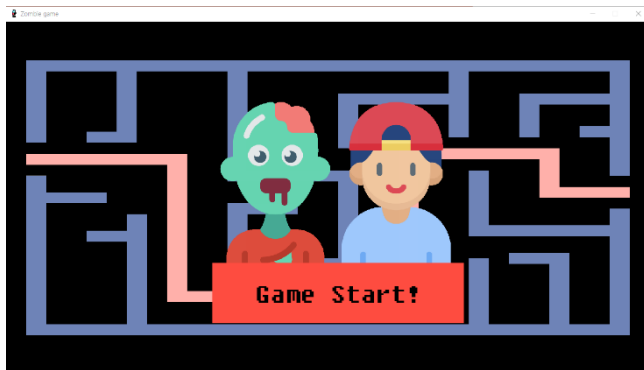


## 3.2 주요 개발 항목의 실적

### ● 4. 게임 맵, 아이템, 좀비와 플레이어 AI를 그래픽으로 구현

#### 1. 게임과 유사한 GUI 구성

- 게임을 실행하면 나타나는 시작 화면
- 세대 종료 시 다음 단계로 넘어가는 화면
  - 달성한 현재 세대와 넘어갈 다음세대 표시
  - 맵 난이도 변경 시 난이도 표시



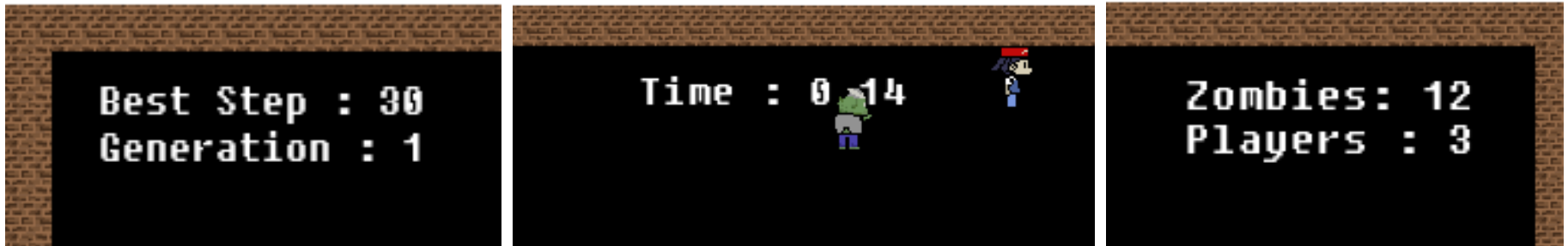
## 3.2 주요 개발 항목의 실적

### ● 4. 게임 맵, 아이템, 좀비와 플레이어 AI를 그래픽으로 구현

#### 1. 게임과 유사한 GUI 구성

##### ■ 현재 점수를 보여주는 전광판

- ◆ best Step과 세대, 게임진행 시간, 필드 내 좀비, 플레이어 수 출력



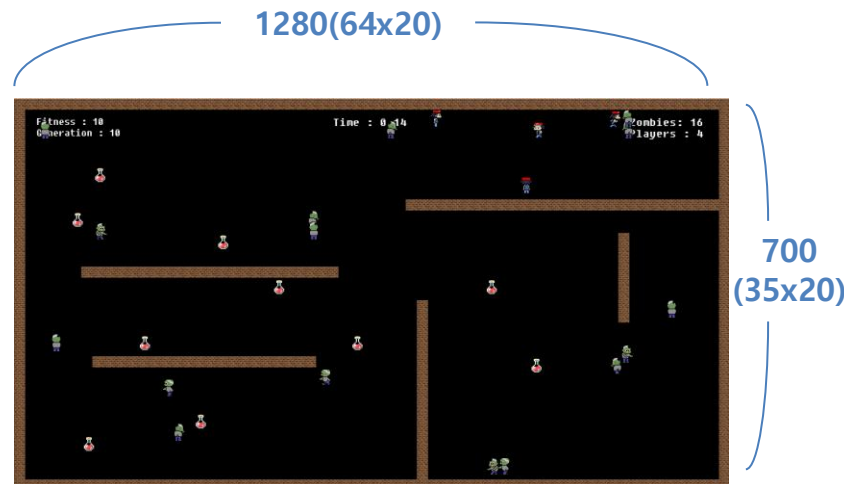
## 3.2 주요 개발 항목의 실적

### ● 4. 게임 맵, 아이템, 좀비와 플레이어 AI를 그래픽으로 구현

#### 2. 게임 내 정적 요소

- 플레이어가 먹는 아이템 그래픽(32x32)
- 맵을 구성하는 벽 그래픽(20x20)
  - ◆ 게임 필드의 크기는 너비 64, 높이 35로 지정
  - ◆ 실제 게임 화면은

(게임 필드 가로길이x벽 그래픽 가로 길이) x (게임 필드 세로길이x벽 그래픽 세로 길이)인 1280x700으로 설정하였다.



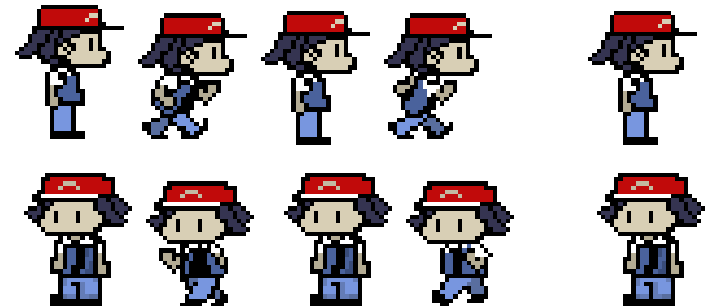


## 3.2 주요 개발 항목의 실적

### ● 4. 게임 맵, 아이템, 좀비와 플레이어 AI를 그래픽으로 구현

#### 3. 게임 내 동적 요소

- 움직이는 플레이어/좀비 그래픽
- 플레이어/좀비는 총 8방향으로 움직인다
- 걷는 애니메이션을 자연스럽게 만들기 위한 최소 프레임은 **3프레임**이 필요
  - ◆ 8방향 x 3프레임 = 이미지 총 24장
- 각 프레임을 0, 1, 2, 3으로 지정하고 클래스 변수로 인덱싱
  - ◆ 방향이 바뀌어도 자연스럽게 걷도록 함



플레이어 애니메이션



좀비 애니메이션

## 3.3 주요 개발 항목의 향후 계획

- 1. 세대를 거칠수록 플레이어AI의 생존시간이 길어지는 유전 알고리즘 구현
  - 적합도 함수 설정
    - ◆ 유전 시키려는 특성을 단순화 하여 빠른 시간 내에 학습 효과를 확인할 수 있도록 함수를 수정&최적화
    - ◆ 개체 적합도를 잘 반영할 함수를 만들어야 함
      - ▶ 제안서에서 서술한 1. 아이템 접근, 2. 좀비 기피, 3. 벽 끼임 기피 중 3번 벽 끼임 기피의 경우, 오히려 플레이어들이 좀비를 피하고자 벽의 구석과 모서리로 몰리는 경향을 발견
    - ◆ 앞으로의 과제
      - ▶ 유전시키려는 형질 간의 충돌 문제가 발생한 것으로 보임
      - ▶ 함수를 좀 더 단순하게 만들어 생존에만 집중할 수 있게끔 함

## 3.3 주요 개발 항목의 향후 계획

- 2. 단계별로 학습한 AI의 생존시간이 그렇지 않은 AI 생존시간의 130% 이상 달성
  - 단계적으로 학습한 플레이어AI와 한가지 맵(1단계)에서 학습한 플레이어AI의 생존시간 비교
    - ◆ 특정 세대에서 전체 플레이어AI의 평균 스텝을 비교
- 3. 좀비가 장애물을 인지하며 플레이어AI를 쫓는 최단거리 탐색
  - 현재 다수의 좀비가 플레이어를 쫓을 시 경로에 대한 연산이 많아지면서 연산 시간이 급격히 증가하는 문제 발생
    - ⇒ 효율화 작업 필요

# 4. 결론

---

4.1 달성 목표

4.2 연구/개발의 의의

4.3 레퍼런스

## 4.1 달성 목표

1. 세대를 거칠수록 플레이어AI의 생존시간이 길어지는 유전 알고리즘 구현
2. 단계별로 학습한 AI의 생존시간이 그렇지 않은 AI 생존시간의 130% 이상 달성
  - 총 4단계의 맵 구성(아무것도 없는 맵, 일자형 벽이 있는 맵, L자 모양 벽이 있는 맵, ㄷ자 모양 벽이 있는 맵)
  - 다양한 맵에서 학습한 플레이어AI와 한가지 맵에서 학습한 플레이어AI의 생존 시간 비교
3. 좀비가 장애물을 인지하며 플레이어AI를 쫓는 최단거리 탐색 알고리즘 구현
  - 맵 단계에 따라 장애물이 추가되어도 플레이어AI를 정상적으로 쫓아간다
4. 게임 맵, 아이템, 좀비와 플레이어 AI를 그래픽으로 구현

## 4.2 연구/개발의 의의

- 본 프로젝트는 **환경 변화**라는 외부 요소를 추가하여 보편적 환경에 적응하는 개체를 만드는 것을 목표로 한다. 이를 통해 단계적 환경 변화가 유전 알고리즘에 성능을 개선시키는 효과를 줄 것이라고 기대할 수 있다.
  - 단순한 맵에서만 학습한 플레이어 AI보다 환경변화를 겪은 플레이어 AI가 더 나은 생존 능력을 보일 것이다.
  - 복잡한 맵에서만 학습한 플레이어 AI보다 환경변화를 겪은 플레이어 AI가 더 짧은 학습시간을 보일 것이다.

## 4.3 레퍼런스

- 유전 알고리즘 및 인공지능경망
  - 사이토 고키, 『 밑바닥부터 시작하는 딥러닝 』, 개앞맵시 옮김, 한빛미디어(2019)
  - 문병로, 『 유전 알고리즘 』, 두양사(2003)
  - 이제영, 고진영, "유전자 알고리즘에 의한 우수 유전자형 선별", 한국데이터정보과학회지 (2009년 7월 7일)
  - 김지민, 김선정, 홍석민, "유전 알고리즘을 이용한 플레이어 적응형 몬스터 생성 기법", 인터넷정보학회논문지 18권2호, 43-51 (2017)

## 4.3 레퍼런스

- A\* 알고리즘
  - Amit's A\* Pages, "astar algorithm", <http://theory.stanford.edu/~amitp/GameProgramming/>, 2020.04.03
  - Develop 마이닝/알고리즘, "Astar", <https://itmining.tistory.com/66>, 2020.04.05
  - ActiveState, "Astar" <http://code.activestate.com/recipes/578919-python-a-pathfinding-with-binary-heap/>, 2020.04.10
  - Youtube, "Astar", <https://www.youtube.com/watch?v=ob4falum4kQ>, 2020.03.27
  - 솔라리스의 인공지능 연구실, "Astar", <http://solarisailab.com/archives/8>, 2020.03.27
  - 생각하는 개발자, "Astar", <http://choi98772.blogspot.com/2014/05/a.html>, 2020.04.10



## 4.3 레퍼런스

- pygame을 이용한 게임개발
  - 알 슈베이가르트, 『 Python과 Pygame으로 게임 만들기 』, 김세희 옮김, 정보문화사(2014)
  - (전반적인 참고)Pygame Documentation", [pygame.org/docs](http://pygame.org/docs), 2020.04.08