2020 프로그래밍 실습I 프로그램개발 보고서

Turtle과 유전이론을 이용한 미로 게임 제작

작성자 : 2학년 2반 3번 이름: 김석준

요약

최근 매체간의 전환이 활발해지면서 문학과 영화, 게임의 경계가 모호해지는 현상이 두드러지고 있다. 그러나, 소설 또는 영화에서 파생된 게임들을 분석해보면 대체로 게임 자체의 아이디어성은 떨어지는 특징이 있었다. 뿐만 아니라. 보다 창의적이고 교육적인 측면에서 이를 만족하는 게임의 비율은 소설 및 영화 파생 여부에 관계없이 점차 감소하는 추세이다. 이에 본 연구자들은 소설 "메이즈 러너"에서 아이디어를 얻어 유전이론을 적용한 미로 게임을 제작하였다.

본 프로그램은 기존의 정적인 미로게임과는 다르게, 유전 이론을 기반으로 한 계속되는 미로의 변경을 일으키는 기능을 포함한다. 이에 사용자는 본 게임을 진행하면서 유전 이론에 기초한 추론능력을 통해 전략적으로 탈출을 할 수 있게 된다.

특히, 본 프로그램은 터틀(Turtle)모듈을 활용하여 빠르고 간편한 드로잉을 통해 프로그램의 효율과 디자인을 확립시켰다는 점이 주요 기술적 요소이다.

물론 지속적으로 미로를 삭제 및 생성시키는 과정에서 방향과 길이를 하나하나 설정하는 것을 설계하고 이해하는 것과 오로지 지식에만 의존하여 유전 이론을 코드로 구현하는 과정에서 어려움을 느꼈으나, 이를 그림으로 나타내고 친구와 계속해서 상의함으로 단계적으로 해결하고 노력하여 프로그램을 완성할 수 있었다.

Ⅰ. 프로그램 개발 동기 및 목적

최근 베스트 셀러가 영화화되고, 영화가 게임화되는 등 다양한 매체간의 상호작용이 활발하게 드러나면서 각 엔터테인먼트간의 경계가 모호해 지고 있다. 이중 뛰어난 줄거리로 많은 사람들의 이목을 집중시킨 소설 "메이즈 러너"는 영화와 게임 등 다양한 매체로써 전환되었다. 그러나, 이중 게임을 살펴보면 기존에 존재하는 게임과 매우 유사하고 소설에서 나타나는 고유한 특징을 반영하지 않은 의미없는 요소가 주를 이루었다. 이는 소설 "메이즈 러너" 뿐 아니라 다른 많은 원작들을 배경으로 한 게임 역시 독특한 특색을 보이는 경우는 매우 드물었다. 이에 본 연구자는 "메이즈 러너"의 지속적인 벽 생성 및 소멸을 구현한 게임을 제작하고자 하였으며, 특히 "유전 알고리즘"을 도입하여 보다 교육적인 방향으로 설계되도록 유도하였다. 본 연구를 통해 유전의 원리를 이용하여 미로를 탈출하는 참신하고 전략적인 새로운 게임을 창출할 수 있었으며, 동떨어진 것처럼 느껴지는 유전 알고리즘을 게임에 적용하는 창의적인 접근을 시도하였다. 본 프로그램으로 유전학에 대한 이해를 게임을 통해 할 수 있도록 유도함으로 교육적 차원에서와 게임의 측면 모두 새로운 방향성을 제시하고자 한다.

Ⅱ. 프로그램 소개

본 프로그램은 정적인 형태의 기존 미로게임의 틀에서 벗어나 보다 동적이고 전략적인 새로운 형태의 미로게임으로, 유전 이론을 바탕으로 하고 있다. 본 게임은 미로를 나타낼 수 있는 16개의 개체가 개체군 내에서 교배하여 다음 세대의 개체군을 형성하는 순환적 구조를 이루고 있으며, 이 때문에 사용자는 참고자료를 참조하여 적절한 전략을 취할 수 있다.

게임을 시작하면 낮부터 시작되며, 하루에는 최대 6번의 이동이 가능하다. 다음날이 될 때마다 미로는 유전 알고리즘을 기반으로 다음 미로를 구성한다. 최종적으로 사용자는 출구를 통해 탈출할 수 있으며, 탈출할 경우 프로그램은 종료된다.

Ⅲ. 메뉴 구성도

Figure1. 메뉴 구성도

메뉴에는 다음과 같은 항목들이 존재한다:

1. 게임하기 #게임을 시작

-어떻게 움직입니까? #이동 방향

2. 게임 규칙 #게임 규칙 설명
3. 종료 #프로그램 종료

Ⅳ. 프로그램 사용법 및 주요 실행 결과

다음은 본 유전 이론을 적용한 미로 게임을 실행하였을 때 데이터 입력과 출력 과정을 나타낸 것이다. 초기 입력은 게임 시작 및 게임 규칙, 종료가 있으며, 게임을 시작할 시 이동 방향에 대한 5가지 입력 방식이 존재한다. 이를 토대로 미로를 전략적으로 빠르게 빠져나가는 것이 이 게임의 주 목적이다.

Figure2. 메인 화면

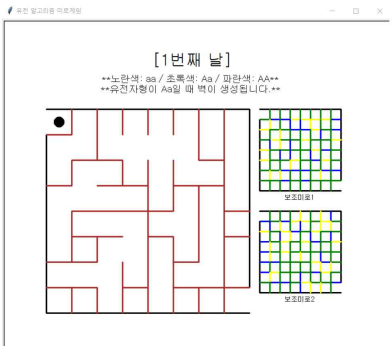

Figure3. 게임 화면

Ⅴ. 프로그램의 차별성

기존의 영화에서 본 뜬 게임들은 배경과 소리 등의 극히 일부 부분만 영화와 비슷하고 나머지 부분은 잘 알려진 식상한 형태의 게임과 매우 유사하다. 그러나, 본 프로젝트에서는 소설 및 영화에서 나타나는 고유 특징들을 적극 반영하여 독특한 게임을 제작할 수 있었다. 특히, 다양한 미로 게임들을 분석했을 때 대체로 정적이고 어려운 미로들을 이용하였으나, 본 미로 게임에서는 계속해서 형태가 바뀌고 유전 이론에 기초한 체계적 전략을 요구하므로 활용 범위와 교육성이 더 높음을 알 수 있다. 어려워 보이는 유전 이론을 미로 게임에 접목시키는 창의적이고 신선한 아이디어로 응용 가능성을 넓혔다는 점에서 프로그램의 차별성이 두드러진다고 할 수 있다.

Ⅵ. 기술적 요소

본 프로그램은 유전알고리즘을 바탕으로 특정 횟수만큼 이동했을 때 정해진 규칙에 따라 미로가 변형되는 특징이 있다. 이때 변형될 미로의 형태는 주 미로의 오른쪽에 있는 두 미로를 통해 유추할 수 있으며, 사용자는 이를 참고하여 미로를 탈출할 수 있다.

가. 유전 알고리즘(heredityAlgorithmpy)

본 유전 알고리즘 파일에서는 미로의 유전 알고리즘 기반 생성 및 소멸에 규칙성을 부여하는 알고리즘이 소개되어 있으며, 이는 개체군 집단의 16마리 개체를 대상으로 진행된다.

아래의 모식도는 미로의 생성 및 변경에 직접적으로 이용되는 개체군(population)에 대한 개념을 나타낸다. 한 세대는 총 16마리의 개체(individual)로 이루어져 있으며, 각 개체는 미로 하나를 나타낼 수 있다. 각 미로 개체는 한 쌍의 염색체(chromosome)가

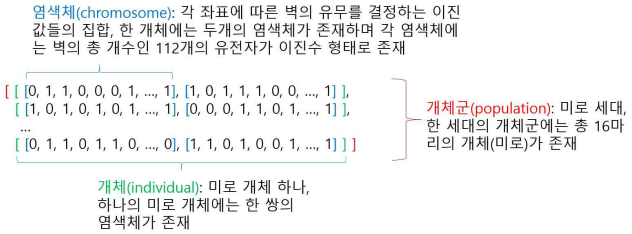

Figure4. 개체군(population) 리스트의 구조

존재하는데, 각 염색체의 같은 인덱스 값에 존재하는 이진값을 비교하여 벽의 존재 유무에 대한 표현형을 결정한다. 따라서, 한 개체군의 리스트는 총 삼중으로 이루어진다.

미로 개체는 이러한 동 세대의 16개의 개체들간의 교배로 다음 세대로 이어지며, 미로의 재배치과정은 다음과 같은 순환과정을 거친다.

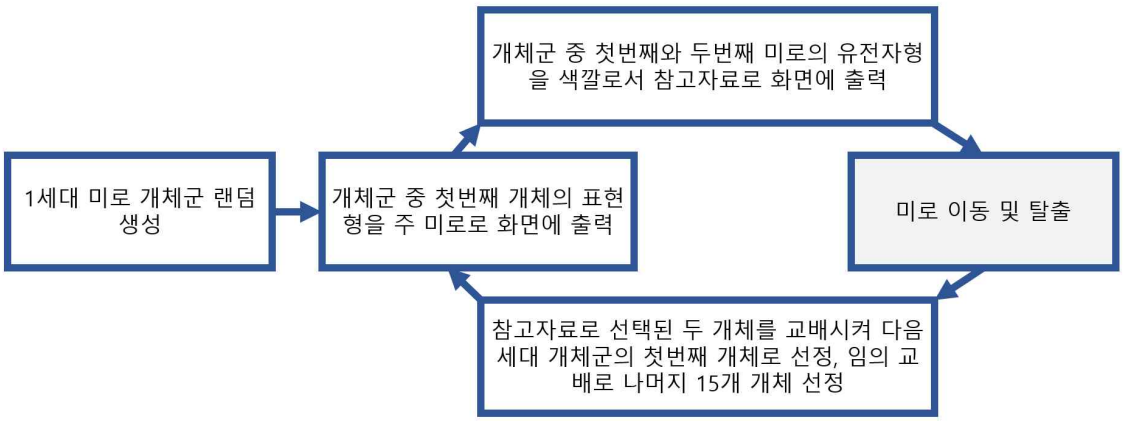

Figure5. 미로 재배치 cycle

우선 랜덤으로 1세대 미로 개체군을 생성한다. 이때, 생성된 16개의 개체 중 첫 번째 개체(population[0])와 두 번째 개체(population[0])의 유전자형을 오른쪽의 보조 미로에 표시한다. 이때, 유전자형에 따라 사용자가 식별할 수 있도록 색깔로써 구분한다. 주 미로는 첫 번째 개체에 대해 표현형을 따라 벽의 유무를 판별하며 생성한다. 이후 6번의 턴이 지나면 보조 미로의 두 미로 개체를 대상으로 교배를 진행하여 자손을 그 다음 주미로에 표현형 형태로 표기한다. 이는 다음 세대 개체군의 첫 번째 개체가 되며, 나머지 15마리의 개체는 이전 세대 개체군 중 임의로 두 마리를 선정해 교배시켜 얻은 자손으로 구성한다. 이후의 과정은 본 알고리즘을 따라 연속적으로 진행된다.

미로는 8*8 형태로 하였으며, 각 미로는 총 112개의 벽에 대한 유전자를 갖는다. 따라서, 염색체 리스트는 112개의 이진 원소로 이루어져있으며, 개체가 갖는 두 염색체의 같은 인덱스 값에 해당하는 이진 유전자간의 비교를 통해 벽의 생성 유무와 같은 표현형을 결정한다. 이때, 미로를 탈출할 때 참고자료로 개체 두 개가 주어지는데, 하나는 현재 화면에 표시된 주 미로의 유전자형이고, 다른 하나는 개체군 리스트 중 두 번째 개체의 유전자형이다. 유전자형은 색깔로써 구분되며, 사용자는 이 두 개체가 교배하여 다음 세대의 개체가 화면에 표시될 것을 통해 미로의 벽 배치를 미리 유추할 수 있다.

이러한 원리를 바탕으로 설계한 유전 알고리즘 함수는 다음과 같다.

```
createNewRandomChromosome(): create new random chromosome;
                                    return chromosome;
createNewRandomPopulation(): create new random population;
                                  return populatoin;
createNextPopulation(old): breeding(old[0], old[1]);
                  for(i=0; i<15; i+=1) do breeding(old[random], old[random]);
                                 return newPopulation;
breeding(ind1, ind2): breed newIndividual between ind1,ind2;
                              return newIndividual;
selectIndividual(population): choose one individual except population[0];
                                  return individual;
phenotype(individual): return list of phenotype of individual;
```

나. 미로 알고리즘(maze.py)

본 프로그램은 일정 횟수만큼 이동하면 미로가 변화되는 동적 게임이다. 이를 기반으로 하는 것은 유전 알고리즘으로, 유전 알고리즘을 통해 제시된 개체들을 바탕으로 미로를 재배치하고 사용자에게 그 배치 알고리즘을 유전 이론을 통해 간접적으로 알려주어 보다 전략적인 미로의 탈출을 진행할 수 있도록 유도하였다.

하루에 6번의 이동이 가능하며, 이동이 모두 끝나면 참고자료로 제시된 미로 개체간의 교배로 나온 자손이 다음 날의 주 미로를 구성한다. 다음은 설계한 미로 알고리즘의 함수이다.

```
makeMazeShape(): print shape of maze;
makeNewMaze(list): print walls of maze based on list;
changemazesub1(individual1): change submaze1 to individual1;
changemazesub2(individual2, count): change submaze2 to individual2;
setcolor(first, second): if first=0 and second=0 then return color1;
                         else if first != second then return color2;
                         else return color3;
changemaze(list): change maze to list;
checkright(x, y, movedirect, list): if the movement available then return 1;
                                    else return 0;
```

Ⅶ. 개발 과정 및 소감

시뮬레이션기, 계산기 등 쉽게 볼 수 있는 기능을 구현한 타 작품과는 달리, 생물학적 지식을 바탕으로 알고리즘을 구현하여 이를 미로라는 별개의 소재와 접목시키는

창의적인 시도를 하였다는 점에서 특이점이 나타난다 할 수 있다. 특히, 유전자형 및 표현형, 우열의 법칙과 같이 단순 공부로만 깨우칠 수 있었던 유전의 이론을 미로의 벽 생성 또는 소멸을 추측하는 사고 과정으로 치환시키면서 이해를 도울 수 있는 프로그램을 스스로 구현하였다는 것에서 문제해결력을 기를 수 있었다.

물론 나의 생물학적 지식을 프로그램으로 구현하는 과정에서 생각을 정리하고 몇몇 규칙을 설정하면서 어려움을 느낀 것이 없지 않아 있었지만, 지식을 정리하고 융합적 사고력을 키울 수 있었다는 점에서 정보학적 사고력을 키울 수 있어 좋았다.

또한, 친구와 함께 프로그래밍을 진행하면서 자신의 생각을 친구에게 전달하고, 특히 친구가 어려워하는 생물학적 개념을 설명하고 구현의 차이를 조율하면서 빠르고 완성도 높은 합작품을 만드는 일련의 과정을 통해 소통의 능력과 공동연구의 중요성을 깨우치게 되었다.

한가지 아쉬운 점은 우리가 개발한 프로그램이 파이썬 상에서 turtle 모듈에 적합하다 판단하여 이를 사용하였으나 turtle의 그리는 속도에 의한 한계가 나타나 시간이 지체된다는 점이었다. 이에 pygame과 같이 프로그램의 성격에 맞으면서 빠르고 편리한 모듈을 알게 된다면 이를 활용하여 다시 프로그램을 설계해보고자 한다.

Ⅷ. 소스 코드

#heredityAlgorithm

```
import random as r

def createNewRandomChromosome():     #랜덤 염색체 생성
    newRandomChromosome = []
    for i in range(112):
        newRandomChromosome.append(r.randrange(2))
    return newRandomChromosome

def createNewRandomPopulation():     #랜덤 개체군 생성
    newRandomPopulation = []
    for i in range(16):
        newIndividual = []
        for j in range(2):
            newIndividual += [createNewRandomChromosome()]
        newRandomPopulation += [newIndividual]
    return newRandomPopulation

def createNextPopulation(oldPopulation):     #다음세대 개체군 생성
    nextPopulation = []
    surviveIdxList=[]
    mainIndividual = breeding(oldPopulation[0], oldPopulation[1])
    nextPopulation+=[mainIndividual]

    oldIdxList = [i for i in range(16)]
    for i in range(15):
        newone = []
        parent = r.sample(oldIdxList, 2)
        father = oldPopulation[parent[0]]
        mother = oldPopulation[parent[1]]
        newone = breeding(father, mother)
        nextPopulation.append(newone)
    return nextPopulation

def breeding(individual1, individual2): #두 개체를 인자로 받은 후 교배시킨 새로운 개체 반환
```

```
    newIndividual = []
    newIndividual.append(individual1[r.randrange(2)])
    newIndividual.append(individual2[r.randrange(2)])
    return newIndividual

def selectIndividual(population):    #미로 개체군 속 처음을 제외한 15개의 미로 중 하나의 미로 선택
    selectedIndividual=[]
    randomSelectIdx = r.randrange(1, 16)
    selectedIndividual = population[randomSelectIdx]
    return selectedIndividual

def phenotype(individual):  #개체의 표현형 리스트 반환
    pheno=[]
    for i in range(112):
        decide = 0
        for chromosome in individual:
            if chromosome[i] == 1:
                decide += 1
        if decide == 1:
            pheno.append(1)
        else:
            pheno.append(0)
    return pheno
```

#maze.py

```
import turtle as t
import heredityAlgorithm as ha

mainx = -100; mainy = -50
subx1 = 120 ; suby1 = 150 ; subx2 = 120 ; suby2 = -50

t.speed(10000)
def makeMazeShape(): #미로 틀 생성
    t.pensize(3)
    t.color("black")
    t.penup()
    t.goto(mainx-200, mainy+200)
    t.pendown()
    t.forward(400)
    t.right(90)
    t.forward(350)
    t.penup()
    t.forward(50)
    t.pendown()
    t.right(90)
    t.forward(400)
    t.right(90)
    t.forward(350)
    t.right(180)
    t.penup()
    t.goto(subx1, suby1)
    t.pendown()
    t.left(90)
    t.forward(160)
    t.right(90)
    t.forward(140)
    t.penup()
    t.forward(20)
    t.pendown()
```

```
    t.right(90)
    t.forward(160)
    t.right(90)
    t.forward(140)
    t.right(180)
    t.penup()
    t.goto(subx2, suby2)
    t.pendown()
    t.left(90)
    t.forward(160)
    t.right(90)
    t.forward(140)
    t.penup()
    t.forward(20)
    t.pendown()
    t.right(90)
    t.forward(160)
    t.right(90)
    t.forward(140)
    t.right(180)
    t.penup()
    t.goto(170, -30)
    t.write("보조미로1", font = ("배달의민족 한나체 Pro 보통", 10))
    t.goto(170, -230)
    t.write("보조미로2", font = ("배달의민족 한나체 Pro 보통", 10))
    t.goto(-190, 200)
    t.write("**노란색: aa / 초록색: Aa / 파란색: AA**", font = ("배달의민족 한나체 Pro 보통", 14))
    t.goto(-192, 180)
    t.write("**유전자형이 Aa일 때 벽이 생성됩니다.**", font = ("배달의민족 한나체 Pro 보통", 14))


def makeNewMaze(list):
    t.color("brown")
    t.penup()
    x = mainx-200
    y = mainy+200
    nownum = 0
    for k in range(8):
        for i in range(7):
            x = x+50
            if(list[nownum+i] == 1):
                t.goto(x, y)
                t.pendown()
                t.forward(50)
                t.penup()
        nownum += 7
        if k == 7:
            break
        y = y-50
        x = mainx-200
        t.left(90)
        for i in range(8):
            if(list[nownum+i] == 1):
                t.goto(x, y)
                t.pendown()
                t.forward(50)
                t.penup()
            x += 50
        nownum += 8
        x = mainx-200
        t.right(90)
    t.color("black")
```

```
def changemazesub1(individual1) :
    t.color("brown")
    t.penup()
    x = subx1; y = suby1
    nownum = 0
    for k in range(8):
        for i in range(7):
            x = x+20
            t.color(setcolor(individual1[1][nownum+i], individual1[0][nownum+i]))
            t.goto(x, y)
            t.pendown()
            t.forward(20)
            t.penup()
        nownum += 7
        if k == 7:
            break
        y = y-20
        x = x-140
        t.left(90)
        for i in range(8):
            t.color(setcolor(individual1[1][nownum+i], individual1[0][nownum+i]))
            t.goto(x, y)
            t.pendown()
            t.forward(20)
            t.penup()
            x += 20
        nownum += 8
        x = x-160
        t.right(90)
    t.color("black")

def changemazesub2(individual2) :
    t.color("brown")
    t.penup()
    x = subx2; y = suby2
    nownum = 0
    for k in range(8):
        for i in range(7):
            x = x+20
            t.color(setcolor(individual2[1][nownum+i], individual2[0][nownum+i]))
            t.goto(x, y)
            t.pendown()
            t.forward(20)
            t.penup()
        nownum += 7
        if k == 7:
            break
        y = y-20
        x = x-140
        t.left(90)
        for i in range(8):
            t.color(setcolor(individual2[1][nownum+i], individual2[0][nownum+i]))
            t.goto(x, y)
            t.pendown()
            t.forward(20)
            t.penup()
            x += 20
        nownum += 8
        x = x-160
        t.right(90)
    t.color("black")
```

```
def setcolor(first, second) :
    str1 = "yellow"
    str2 = "green"
    str3 = "blue"
    if first == 0 and second == 0:
        return str1
    if first != second :
        return str2
    if first == 1 and second == 1:
        return str3

def changemaze(list):
    t.color("brown")
    t.penup()
    x = mainx-200
    y = mainy+200
    nownum = 0
    for k in range(8):
        for i in range(7):
            x = x+50
            if(list[nownum+i] == 1):
                t.goto(x, y)
                t.pendown()
                t.forward(50)
                t.penup()
        nownum += 7
        if k == 7:
            break
        y = y-50
        x = mainx-200
        t.left(90)
        for i in range(8):
            if(list[nownum+i] == 1):
                t.goto(x, y)
                t.pendown()
                t.forward(50)
                t.penup()
            x += 50
        nownum += 8
        x = mainx-200
        t.right(90)
    t.color("black")

def checkright(x, y, movedirct, list) :
    placex = int((-mainx+x+200)/50)
    placey = int(((-mainy+y)*(-1) + 200)/50)
    if movedirct == '1':
        if x == mainx+175 or list[placex + placey*15] == 1:
            return 0
    if movedirct == '2':
        if x == mainx-175 or list[placex+placey*15-1] == 1:
            return 0
    if movedirct == '3':
        if y == mainy+175 or list[placex+placey*15-8] == 1:
            return 0
    if movedirct == '4':
        if y == mainy-175 or list[placex+placey*15+7] == 1:
            return 0
    return 1

def main():
```

```
t.title("유전 알고리즘 미로게임")
t.hideturtle()
t.color("black")
t.penup()
t.goto(-240, 70)
t.write("유전 알고리즘 미로게임", font = ("배달의민족 한나체 Pro 보통", 36, "bold"))
t.goto(-140, 30)
t.write("시행을 Shell에 입력해주세요", font = ("배달의민족 한나체 Pro 보통", 18, "bold"))
t.goto(-60, -10)
t.write("1. 게임하기", font = ("배달의민족 한나체 Pro 보통", 16))
t.goto(-60, -40)
t.write("2. 게임 규칙", font = ("배달의민족 한나체 Pro 보통", 16))
t.goto(-60, -70)
t.write("3. 종료", font = ("배달의민족 한나체 Pro 보통", 16))

print("===유전 알고리즘 미로게임===")
while 1:
    print("1. 게임하기")
    print("2. 게임 규칙")
    print("3. 종료")
    choice = int(input("입력: "))
    if choice == 1:
        break
    elif choice == 2:
        print()
        print("게임규칙은 다음과 같습니다.")
        print("[미로]")
        print("1. 러너(runner)는 하루에 6번씩만 미로의 한 칸을 이동할 수 있습니다.")
        print("2. 밤이 끝날 때마다 미로는 재배치됩니다.")
        print("3. 최대한 빨리 미로를 탈출해야 합니다.")
        print("[전략]")
        print("1. 오른쪽 보조미로는 두 미로의 유전자형을 색깔로 나타낸 것이며, 위의 보조미로가
현재 주 미로입니다.")
        print("2. 미로의 유전자형에는 AA, Aa, aa가 있으며, 각각 파란색, 초록색, 노란색으로
나타납니다.")
        print("3. 유전자형이 Aa인 개체만 벽이 나타납니다.")
        print("4. 미로의 재배치는 오른쪽 보조 미로 2개의 교배로 나타납니다.")
        print("5. 보조미로의 유전자형을 통해 다음 미로의 배치를 예상할 수 있습니다.")
        input("아무거나 입력하면 메뉴로 돌아갑니다.")
        print()
    elif choice == 3:
        print()
        print("게임을 종료합니다.")
        return
    else:
        print("다시 입력해주세요.")

print()
print("===게임 시작===")

t.clear()
t.bgcolor("gray")
t.shape("circle")
t.hideturtle()
makeMazeShape()

pop = ha.createNewRandomPopulation()
individual = ha.selectIndividual(pop)
pheno = ha.phenotype(pop[0])
makeNewMaze(pheno)
changemazesub1(pop[0])
changemazesub2(individual)
```

```
movecnt = 0
day=0
x = mainx-175
y = mainy+175

while 1 :
    t.goto(x, y)
    t.showturtle()

    if movecnt%6==0:
        day+=1
        t.goto(-90, 230)
        t.write("[%d번째 날]" %day, font = ("배달의민족 한나체 Pro 보통", 24))
        t.goto(x,y)
        print()
        print("***%d번째 날이 밝았습니다.***" %day)
        print()
        print("[아침 8시]")
        t.bgcolor("white")
    elif movecnt%6==1:
        print()
        print("[낮 12시]")
    elif movecnt%6==2:
        print()
        print("[오후 4시]")
    elif movecnt%6==3:
        print()
        print("[저녁 8시]")
        t.bgcolor("gray")
    elif movecnt%6==4:
        print()
        print("[밤 12시]")
    elif movecnt%6==5:
        print()
        print("[새벽 4시]")

    while 1:
        n = input("어떻게 움직입니까? 1.오른쪽 / 2.왼쪽 / 3.위쪽/ 4.아래쪽 / 5.움직이지 않음")
        if n != '1' and n != '2' and n != '3' and n != '4' and n!='5':
            print("다시 입력해주세요.")
            print()
            continue

        if checkright(x, y, n, pheno) == 0:
            print("움직임이 불가능합니다. 다시 입력해주세요.")
            print()
            continue
        else:
            if n == '1':
                x += 50
            if n == '2':
                x -= 50
            if n == '3':
                y += 50
            if n == '4':
                y -= 50
            movecnt += 1
            break

    if x == mainx+175 and y == mainy-175:
        t.goto(x, y)
```

```
            print()

            print("===게임 종료===")
            if (movecnt-1)%6==0:
                print("기록: %d번째 날 아침 8시 탈출" %day)
                return
            elif (movecnt-1)%6==1:
                print("기록: %d번째 날 낮 12시 탈출" %day)
                return
            elif (movecnt-1)%6==2:
                print("기록: %d번째 날 오후 4시 탈출" %day)
                return
            elif (movecnt-1)%6==3:
                print("기록: %d번째 날 저녁 8시 탈출" %day)
                return
            elif (movecnt-1)%6==4:
                print("기록: %d번째 날 밤 12시 탈출" %day)
                return
            elif (movecnt-1)%6==5:
                print("기록: %d번째 날 새벽 4시 탈출" %day)
                return

        if movecnt%6 == 0:
            print()
            print("미로 재배치중...")
            print()
            t.hideturtle()
            pop = ha.createNextPopulation(pop)
            individual = pop[0]
            individualb = ha.selectIndividual(pop)
            pheno = ha.phenotype(pop[0])
            t.clear()
            t.right(270)
            makeMazeShape()
            changemaze(pheno)
            changemazesub1(pop[0])
            changemazesub2(individualb)
            t.goto(x, y)
            t.showturtle()

main()
```