상태 기계 행위

상태기계 : 각 행위가 하나의 상태를 가지는 것

상태 그 자체는 부분적으로 하나의 상태 기계만 정의한다. 그래서 상태 기계를 변경하거나 상태 기계 간 전이(transition).하는 방법을 결정하는 것이 중요하다. 각 상태는 진입하거나 빠져나갈 때 특정 행위를 수행한다. (몇 가지의 상태를 갖는지 계획하고 각 상태의 대한 전이를 정의한다.)

상태를 열거형으로 선언한다. (enum) – 간단하지만 확장성이 부족하다(상태가 많아질수록 update, enter, exit 함수가 많아지고 가독성 떨어짐) ( 여러 AI간에 기능을 혼합하고 일치시키는 것도 쉽지 않다.)

상태를 클래스를 사용해서 나타낸다.

상태 진입시 OnEnter, 상태 종료시 OnExit, AIState 인스턴스 포인터의 해시맵을 가져서 상태를 해시 맵에 추가시킨다.

상태 전이시 OnExit로 상태 종료 후 새로운 상태를 찾고 변경한 뒤 OnEnter로 들어간다. 상태를 찾지 못하면 에러메시지

길찾기

그래프를 이용( 노드, 엣지, 가중치 가 있다.)

너비 우선 탐색 : 에지에 가중치가 없거나 모든 에지가 양의 같은 가중치를 가질 경우 최단 경로를 찾는 것을 보장한다.

BFS 동안에 각 노드는 직전에 방문한 노드를 알아야 한다.(부모노드)

노드 맵을 큐로 사용하여 BFS를 구현한다.

BFS가 성공하면 outMap의 부모 포인터를 사용해서 경로 재구축이 가능하다.

목표노드의 부모노드 이면 목표에서 한칸, 부모노드의 부모노드이면 목표에서 두칸인 셈이다. 하지만 이는 목표점에서 시작점으로의 경로이다. 스택을 사용해서 반전시킬 수 도 있지만 탐색 자체를 반전시키면 된다.

휴리스틱 : 예상되는 결과를 근사하는 함수이다.

H(x) 는 노드 x에서 목표 노드까지의 추정 비용을 뜻한다.

휴리스틱 함수는 노드 x에서 목표까지의 실제 비용보다 같거나 작으면 항상 허용이다. 실제보다 큰 경우가 있다면 허용되서는 안된다.(사용 x)

사각형 격자에서 휴리스틱을 계산하는 데는 두가지 방법이 있다.

첫번째는 대각선 이동은 유효하지 않다고 가정하는 것이다.

H(x) = |start.x – end.x| + |start.y – end.y|

두번째는 유클리드 거리이다. 표준 거리 공식을 사용한다.(일직선 경로)

H(x) =

두번째 휴리스틱 함수가 더 추천할만 하지만 첫번째가 계산이 효율적이다.