공학도를위한창의적컴퓨팅 과제#4

~ 3. 의사 결정 가이드 (2) ~

이 설명서에 대한 설명

- 작업자 스타일에 대한 좀 더 자세한 설명이 적혀 있어요.
  - 수집가 스타일을 고르려는 친구들도 살짝 구경해 봐요
- 부록으로, 게임 안에서 사용되는 각종 Data에 대한 명세를 담아 두었어요. 주제별로 나누어 두었으니 내 캐릭터 모듈 만들 때 참고하면 돼요.
- (중요)일부러 짧게 적어 놨어요. 읽다가 궁금한 점이 생기면 꼭 강사에게 문의해 주세요.

작업자 관련 자세한 설명

이 단원에서는 게임에서 제시하는 두 가지 스타일 중 하나인 작업자 스타일에 집중해서 게임 내 각종 요소들을 자세히 설명합니다. 수집가 스타일 캐릭터를 만들 친구들도 여유가 있다면 '이런 것도 여기에 들어 있구나' 하는 느낌으로 한 번 읽어보는 것을 권장합니다. (수집가 스타일 설명과 다르게 추천 세부 스타일 소개는 없어요. 자유롭게, 자신이 원하는 대로 만들면 돼요)

작업자 캐릭터의 목표(위에 적어 둔 것이 더 중요)

- 수집가 캐릭터들을 위해 '적절'한 칸에 새 거점 건설
  - 수집가 친구들은 보통 거점 주변 두 칸 범위, 가끔 세 칸 범위 안에서 활동할 예정
  - 따라서 새 거점은 이전 거점들과 네 칸 정도 떨어진 칸에 건설하는 것이 적당함
    (멀면 자원 운반이 어려움. 가까우면 수집가 친구들이 새로 수집 가능한 영역이 줄어듦)
  - 건설에는 자원이 다수 필요하므로 기존 거점에 자원이 모인 이후 착수하는 것을 권장
- 기본 목표: 거점에 수납되어 있는 자원을 인출하여 칸 점유에 사용
  - 직접 자원을 수집하면서 점유에 임하는 것은 그리 효율적이지 않을 것임
  - 기본 목표지만 이를 달성하기 위해 자원이 다량 필요하므로 확장할 때는 거점 건설을 먼저 고려하는 것이 효율적
- '적절'한 칸에서 새 요청 게시
  - 일반적인 캐릭터들은 의사 결정 도중에 몇 십 명에 달하는 다른 캐릭터들의 위치나 최근 행동 이력을 조회하지 않으며, 대신 주로 (건설 완료된) 거점 목록을 활용함
  - 따라서, 특히 새 거점을 건설할 때는 해당 칸에 대한 거점 건설 요청, 또는 중심 칸에서 매우 멀리 있는 칸의 경우 텔레포터 건설 요청을 게시해 두는 것을 추천함(이유는 후술)

- 거점간 이동 효율을 높이기 위한 칸 점유 및 텔레포터 건설
  - 기존 거점에 자원이 충분히 많이 수납되어 있는 경우, 전반적인 이동 효율을 높이기 위해 거점들 사이의 칸들을 점유하거나 텔레포터 건설에 착수하는 것을 고려할 수 있음
  - 수집가를 포함한 모든 캐릭터들은 중심 칸에서 게임을 시작하므로. 중심 칸 및 그 주변 거점에는 필연적으로 자원이 다량 수납되어 있게 됨. 텔레포터를 건설해 두면 중심 칸에서 먼 곳까지 확장할 때도 해당 현장에 텔레포터를 건설함으로써 보다 수월하게 자원을 인출할 수 있게 됨
    - 이동 행동과 워프 행동의 기력 소모량 및 후딜레이는 동일
  - 수집가 캐릭터들이 한 거점 주변에서 오랜 시간 활동한다면, 작업자 캐릭터들은 인출을 위해 여러 거점들 사이를 이동하는 빈도가 높음. 따라서 보다 효율적인 자원 유통을 위해 거점과 텔레포터가 모두 건설된 칸에 대해 초과 점유를 진행해 두는 것도 고려해볼 수 있음
    - 초과 점유는 워프 행동의 후딜레이도 감소시킴
  - 텔레포터가 건설되기 전에도 작업자들은 각 거점들 사이를 직접 이동하며 자원을 확보하게 될 가능성이 높음. 따라서 두 거점들 사이의 최단 경로를 구성하는 칸들을 우선적으로 점유해 둠으로써 작업자들과 '효율을 신경 쓰는 수집가들'의 관심을 유도할 수 있음
  - **(중요)**중심 칸과의 거리에 따라 건설에 필요한 자원량이 증가함(세부 수치는 부록 참조)
    - 중심 칸 주변에서는 거점은 50, 텔레포터는 100 필요
    - 중심 칸과의 거리가 256일 때 둘 다 122 필요
    - 거리가 512일 때 거점은 300(중심 칸 주변 대비 6배), 텔레포터는 150(3배) 필요
    - 작업자들은 이 점을 감안하여 다음 확장 방향 및 그 순서를 결정할 필요가 있음
- 마지막으로, 위에 나열된 각 목표들을 달성하기 위한 Code들을 다수 마련해 두고, 현재 상황에 따라 여럿 중 하나를 골라 수행할 수 있도록 분기 흐름을 구성해 두어야 함
  - 수집가들은 자신의 개인 평면에 집중하여 의사 결정을 수행하므로 Data의 변화를 예측하기 수월하며, 다른 캐릭터나 공유 평면에 대해 그리 신경 쓰지 않아도 무방함
  - 반면 작업자들은 시시각각 변하는 공유 평면을 기반으로 의사 결정을 수행하기 때문에 수집가의 경우처럼 할 일 하나를 정하고 우직하게 이행하는 것이 비효율적이거나 아예 불가능할 수 있음
    - 예를 들어, 이미 거점 건설이 완료된 칸에서 건설 의사 결정을 수행하면 '유효하지 않은 의사 결정'으로 간주됨
  - 따라서, 이전 의사 결정의 결과(행동 이력)를 체크하여 유효성 여부를 확인하거나, 매 의사 결정마다 현재 할 일의 적합성을 확인하여 적절히 전환할 수 있는 Code가 필요함

(중요)작업자 캐릭터의 세부 스타일 – 기본적으로 자유

- 게임에 참여하는 모든 캐릭터들의 기본 목표는 '본 게임'에서 더 많은 칸을 점유하는 것임
- 수집가 캐릭터를 만드는 친구들은 이러한 목표를 좁혀 가며 접근함
  - 직접 점유에 임하는 스타일도 있지만, 보통은 점유에 필요한 자원을 수집하여 거점에 수납하는 것을 자신의 목표로 지정함
  - 주로 활동할 범위(중심 칸과의 거리로 특정되는)를 정함
  - 이렇게 목표를 좁혀 둔 다음 효율성을 챙기기 위해 능력치 조합 및 각종 흐름을 만듦
- 작업자 캐릭터들 또한, 기본 목표를 달성하기 위한 작업들을 모두 혼자 수행할 수는 없으므로, 수행 가능한 여러 작업들 중 하나를 골라 가며 의사 결정을 수행하게 됨
- (중요)작업자가 수행하는 대부분의 작업들은 '본 게임' 이전에는 온전히 테스트해보기 어려움. 작업자들이 주로 다루는 Data는 균등하거나 고정되어 있지 않으며, 여러 캐릭터들의 행동 결과가 누적되어 형성되는 것이기 때문임
  - 제출하기 전에도 캐릭터들을 여럿 추가한 다음 테스트를 진행해 볼 수는 있으나, '본 게임'에 참여할 50여 명의 캐릭터 조합을 미리 구성해 보는 것은 강사도 불가능함
- (매우 중요)따라서 역설적으로, 수집가 캐릭터를 만들 때에 비해서, 그러한 '본 게임에서의 효율성'을 덜 신경 쓰며 소신 있게 자신만의 캐릭터를 구성해도 무방함
  - 일단 '본 게임'에 참여하는 작업자 캐릭터 수는 다섯 명 정도에 불과할 것이므로, 중반 이후 확장이 충분히 진행된 시점부터 작업자들은 서로 다른 영역에서 활동하게 될 것임
  - 그러므로, 자신의 작업을 돕기 위해 따라오는 수집가 캐릭터들을 위해...
    - 중간중간 적절한 요청을 게시
    - 새 거점 또는 텔레포터를 건설할 때는 기존 거점 주변 칸을 고르도록 노력
    - ...하는 정도의 기본 흐름만 갖추어 둔 채 구체적인 흐름은 자유롭게 가져가도 됨
- **(어쩌면 가장 중요)그래도 혹시 '이러다 나 혼자 멀리서 작업하고 있는 거 아닌가' 하고 불안감을 느끼고 있다면, 제출 이전에 미리 오픈톡방 등에 내 의견을 게시하거나 강사에게 살짝 문의하는 것을 적극 권장함**
  - '저는 서쪽으로 일직선으로 확장할래요'와 같은 단순한 스타일이라 하더라도, 이를 미리 여러 친구들에게 언급해 두고 몇몇 친구들에게 '저도 같이 갈래요' 답변을 받을 수 있다면 내 (작업자) 캐릭터의 전반적인 작업 효율을 안정적으로 확보할 수 있음
  - 중간고사 결과 및 지난 과제 제출물들을 알고 있는 강사와 함께 내 캐릭터 스타일에 대해 논의하면서 보편적인 수집가 캐릭터들의 활동을 예상하고 그에 어울리는 세부 의사 결정 흐름을 계획해볼 수 있음

부록 소개

여기서부터 이 설명서의 부록이에요. 부록의 각 단원에는 hw4_core.py에 반영되어 있는 게임 속 요소들에 대한 자세한 내용이 나열되어 있어요(그러니 굳이 Core **모듈**을 직접 열어 구경하지 않아도 돼요). 조금 더 효율적인 캐릭터를 만들고 싶을 때 살짝 참고해 보면 좋을 거예요.

부록#1. 행동

이 단원에는 각 캐릭터들이 매 의사 결정의 결과로서 수행할 수 있는 행동들에 대한 자세한 설명이 적혀 있습니다.

공통

- 캐릭터는 게임 시작 직후(게임 내 시계 기준 0초째)에, 그리고 직전 행동에 대한 후딜레이가 끝난 시점에 의사 결정을 수행하며, 그 결과가 유효한 경우 행동을 수행하게 됩니다.
  - 의사 결정은 한 명씩 수행합니다. 여러 캐릭터가 게임 내 시계 기준 동일한 시점에 의사 결정을 수행하게 된 경우(게임 시작 직후가 대표적), 아래 우선 순위에 따라 순서가 결정됩니다:
    - 직전 의사 결정을 수행한 시점이 더 빠른 캐릭터가 먼저
    - 민첩성 능력치가 더 높은 캐릭터가 먼저
    - 이름이 더 '빠른' 캐릭터가 먼저(최종 버전에서는 먼저 제출한 캐릭터가 먼저)
  - 결과가 유효하지 않거나(행동 실패) 의사 결정 도중 오류를 발생시킨 경우 대기 행동을 선택한 것으로 간주됩니다
- '의사 결정 결과가 유효하지 않음'은 아래와 같은 상황을 의미합니다:
  - 이상한 **값**을 return함
  - 캐릭터가 특정 행동을 수행할 수 없는 상황에서 그 행동을 선택함
    (예: 텔레포터가 없는 칸에서 워프 행동을 선택)
  - 주의: 어떤 행동은 수행할 수는 있지만 실질적인 효과가 없을 수도 있으며, 그 행동을 선택해도 여전히 유효한 의사 결정으로 간주함
    (예: 자원이 없는 칸에서 수집 행동을 선택)
- 행동별 설명에 세부 **수식**을 적을 때는 여러분이 적을 수 있는 **이름**을 사용하여 표시해 두었습니다(가독성을 위해 `infos` **이름 사전**에 등재된 **이름**들은 풀 네임 없이 그냥 그 **이름**을 적어 두었습니다).
- 각 **값**들을 결정하기 위한 세부 **수식**은 부록#3 단원에서 찾아볼 수 있습니다. Ctrl + F를 눌러 검색해 보세요.

0 – 수집

- 개인 평면의 현재 칸에 놓인 자원을 '들 수 있는 만큼' 들어 올립니다.
  - 들 수 있는 만큼: `min(max_r_carrying - r_carrying, myPlane[pos_me])`
- 수집 의사 결정은 항상 유효합니다. 들 수 있는 자원이 하나도 없더라도 실패하지 않습니다.
- 기본 후딜레이: `1.0`초
  - 기력이 `0` 이하인 경우 3배
- 기력 소모량: `10`

1 – 이동

- 캐릭터의 좌표를 지정한 방향으로 한 칸만큼 옮깁니다.
  - `0`: 위(y축 -방향), `1`: 왼쪽(x축 -방향), `2`: 오른쪽(x축 +방향), `3`: 아래(y축 +방향)
- 방향 **값**이 유효하다면 이동 의사 결정은 항상 유효합니다.
- 기본 후딜레이: `1.0`초
  - 현재 칸이 점유 완료되어 있다면 `publicPlane[pos_me].coef_delay_after_move` 배
    - 기력과 무관하게 항상 적용
  - 기력이 `0`보다 큰 경우 `coef_delay_after_move` 배
  - 기력이 `0` 이하인 경우 3배
- 기력 소모량: `max(r_carrying - coef_stamina_cost_reduction_after_move, 1)`

2 – 점유

- 들고 있는 자원을 모두 사용하여 현재 칸에 대한 점유 기여량을 높입니다.
  - 점유 완료까지 필요한 자원량은 `publicPlane[pos_me].r_toOccupy` 로 확인 가능
- 점유 의사 결정은 항상 유효합니다. 들고 있는 자원이 없어도 실패하지 않으며, 점유 완료된 칸에 대해 수행할 수도 있습니다('추가 점유'로 간주).
- 기본 후딜레이: `8.0`초
  - 현재 기력이 `0`보다 큰 경우 `coef_delay_after_work` 배
  - 현재 기력이 `0` 이하인 경우 3배
- 기력 소모량: `10`

3 – 거점 건설

- 현재 칸에 거점이 없다면, 들고 있는 자원을 모두 사용하여 건설 기여량을 높입니다.
  - 건설 완료까지 필요한 자원량은 publicPlane[pos_me].r_toBuildBase 로 확인 가능
- 들고 있는 자원이 없어도 유효하지만, 현재 칸에 이미 거점이 건설되어 있는 경우 실패합니다.
- 기본 후딜레이: 10.0초
  - 기력이 0보다 큰 경우 coef_delay_after_work 배
  - 기력이 0 이하인 경우 3배
- 기력 소모량: 20

4 – 텔레포터 건설

- 현재 칸에 텔레포터가 없다면, 들고 있는 자원을 모두 사용하여 건설 기여량을 높입니다.
  - 완료까지 필요한 자원량은 publicPlane[pos_me].r_toBuildTeleporter 로 확인 가능
- 들고 있는 자원이 없어도 유효하지만, 현재 칸에 이미 텔레포터가 있는 경우 실패합니다.
- 기본 후딜레이: 10.0초 (거점 건설 행동과 동일)
  - 현재 기력이 0 초과인 경우 coef_delay_after_work 배
  - 현재 기력이 0 이하인 경우 3배
- 기력 소모량: 20 (거점 건설 행동과 동일)

5 – 수납

- 현재 칸에 거점이 있다면, 들고 있는 자원을 모두 그 거점에 내려놓습니다.
  - 거점에 있는 자원량은 publicPlane[pos_me].r_stored 로 확인 가능
    (거점이 없을 때 위 **수식**을 **계산**하면 0 나옴)
- 들고 있는 자원이 없어도 유효하지만, 현재 칸에 거점이 없는 경우 실패합니다.
- 기본 후딜레이: 0.5초
  - 현재 기력이 0 이하인 경우 3배
- 기력 소모량: 3

6 – 인출

- 현재 칸에 거점이 있다면, 자원을 '들 수 있는 만큼' 들어 올립니다.
  - 들 수 있는 만큼: `min(max_r_carrying-r_carrying, publicPlane[pos_me].r_stored)`
- 들 수 있는 자원이 하나도 없더라도 유효하지만, 현재 칸에 거점이 없는 경우 실패합니다.
- 기본 후딜레이: `0.5`초
  - 현재 기력이 `0` 이하인 경우 3배
- 기력 소모량: `5`

7 – 워프

- 현재 칸과 지정한 칸에 모두 텔레포터가 있는 경우 캐릭터의 좌표를 지정한 칸으로 옮깁니다.
  - **값**을 두 개 더 return해야 하며, 각각 순서대로 x좌표, y좌표로 간주됨
    (예: **문장** `return 7, 3, 5` 엔터 를 **실행**하면 (3, 5)로 워프하겠다고 정한 셈이 됨)
- 현재 칸 또는 지정한 칸에 텔레포터가 없다면 실패합니다.
- 기본 후딜레이: `1.0`초
  - 현재 칸이 점유 완료되어 있다면 `publicPlane[pos_me].coef_delay_after_move` 배
    - 기력과 무관하게 항상 적용
  - 현재 기력이 `0`보다 큰 경우 `coef_delay_after_move` 배
  - 현재 기력이 `0` 이하인 경우 3배
- 기력 소모량: `max(r_carrying - coef_stamina_cost_reduction_after_move, 1)`

8 – 요청 게시

- 아래에 나열된 조건을 모두 만족하는 경우 현재 칸에 대해 지정한 종류의 요청을 게시합니다.
  - `0`: 자원 수집 요청, `1`: 거점 건설 요청, `2`: 텔레포터 건설 요청
- 조건:
  - 내 캐릭터가 게시했고 아직 달성되지 않은 요청이 존재하지 않아야 함
    (아직 한 번도 요청을 게시한 적이 없거나, 게시한 요청이 이미 달성되어 있어야 함)
  - 거점/텔레포터 건설 요청의 경우 현재 칸에 거점/텔레포터가 없어야 함(있다면 실패)
  - 주의: 이미 자원 수집 요청이 게시되어 있는 칸에 새로운 자원 수집 요청을 게시할 수 있음. 그러나 먼저 게시된 요청부터 취급되기 시작하므로 경우에 따라 새로운 요청이 달성되기까지 시간이 오래 걸릴 수 있음
- 요청하는 자원량
  - 자원 수집 요청: 개인 평면 위 해당 칸에 원래 놓여 있던 자원량의 `10`배만큼
  - 거점/텔레포터 건설 요청: 건설 완료까지 남은 자원량
- 요청 기여 판정
  - 자원 수집 요청: 해당 칸 및 주변 두 칸 범위 안에서 수집 행동을 통해 자원을 들어 올린 만큼 요청 달성에 기여하게 됨
  - 거점/텔레포터 건설 요청: 해당 칸에서 해당 건설 행동을 수행하여 자원을 사용한 만큼 요청 달성에 기여하게 됨
- 기본 후딜레이: `0.5`초
  - 현재 기력이 `0` 이하인 경우 `3`배
- 기력 소모량: `10`

9 – 대기

- 현재 칸에서 대기합니다. 기력이 회복됩니다.
- 후딜레이: `1.0`초 (기력과 무관하게 고정)
- 기력 회복량
  - 현재 칸에 거점이 없다면 `coef_stamina_heal` 만큼 회복
  - 현재 칸에 거점이 있다면 `coef_stamina_heal_on_base` 만큼 회복

부록#2: 점수 기록 시스템

이 단원에는 게임 내에서 여러분이 제출한 캐릭터의 의사 결정들을 어떻게 집계하는지에 대한 서술이 적혀 있습니다. 항목이 조금 많기는 하지만, 스타일마다 중요하게 고려해야 할 항목이 서로 다르고, 일부 항목의 경우 과제 점수 산출에 직접적으로 사용될 예정이니, 최종 버전 **모듈**을 만들기 전에 한 번 구경해 보면 좋겠습니다.

공통

- 내 캐릭터의 현재 점수는 `infos.scores` 를 사용하여 확인할 수 있음.
  - 예: 내 캐릭터가 수집 행동을 수행하며 실제로 적용받은 후딜레이 합을 **계산**하고 싶다면 **수식** `infos.scores[0x20]` 을 적으면 됨
  - (중요)다른 캐릭터의 점수는 알 수 없음
    - 따라서 '다른 캐릭터들보다 조금만 더 해야지'와 같은 목표를 추구하는 것은 어려움
- 점수 **값**의 **형식**은, 보통은 `int` **형식**, 후딜레이 또는 현실 시간과 관련한 **값**은 `float` **형식**임.
- 일부 튜토리얼 시나리오의 경우 해당 시나리오에서 중시하는 점수 **값**들을 게임 종료 이후에 바로 출력하도록 만들어 둠.
  - 원한다면 종료 이후 interactive에서 직접 **수식**을 적어 가며 내 캐릭터의 다른 점수들을 확인해 볼 수 있음
    - 단, 이러기 위해서는 게임을 시작할 때 내 캐릭터 **모듈**을 붙잡고 F5를 눌러야 함

요약 표

아래 표를 보고 index 값을 확인한 다음 Ctrl + F를 눌러 세부 설명을 검색해 보는 것을 추천:

Index 범위	설명	형식	비고
0x00 ~ 0x09	각 행동을 수행한 횟수	`int`	
0x0a ~ 0x0d	의사 결정과 관련한 부가적인 **값**들	다양	
0x10 ~ 0x19	행동별 주요 **값**들을 누적한 것	`int`	과제점수관련O
0x1a ~ 0x1c	자신이 수집한 자원이 어느 작업에 사용되었는지	`int`	과제점수관련O
0x1d ~ 0x1f	점유 및 건설을 통해 다른 캐릭터를 도와 얻은 점수	`int`	과제점수관련O
0x20 ~ 0x29	각 행동을 수행하며 실제로 적용받은 후딜레이 합	`float`	
0x2a ~ 0x2d	점유된 칸에서 이동/워프한 횟수 및 절약한 후딜레이	다양	
0x30 ~ 0x32	수집, 거점 건설, 텔레포터 건설에 기여한 총 자원량	`int`	
0x33 ~ 0x38	힘 능력치와 관련한 효율을 체크하기 위한 **값**들	`int`	
0x39 ~ 0x3e	민첩성 능력치와 관련한 효율을 체크하기 위한 **값**들	다양	
0x3f ~ 0x41	체력 능력치와 관련한 효율을 체크하기 위한 **값**들	다양	

`0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09` – 각 행동을 수행한 횟수: `int`

- 각각 수집, 이동, 점유, 거점 건설, 텔레포터 건설, 수납, 인출, 워프, 요청 게시, 대기를 의미
  - 내 캐릭터 **모듈**에 **수식** 적을 때는 각각 `ret_gather ~ ret_wait`를 사용해도 됨
- 유효한 의사 결정을 수행하여 실제 행동을 수행할 때마다 1씩 증가함.
  - 의사 결정 결과가 유효하지 않거나, 의사 결정 도중 오류를 발생시킨 경우에도 대기 행동을 수행하기는 하지만, 이 때는 후술할 별도의 항목으로 다룸
- 튜토리얼 시나리오에 도전할 때 내 캐릭터 **모듈**의 개략적인 성능을 확인하는 지표로 활용할 수 있음.
  - '본 게임'을 준비하는 단계에서는 횟수보다는 후딜레이 합(사용한 시간)을 줄이기 위해 노력하는 것이 더 유리할 수 있음

`0x0a` – 유효하지 않은 의사 결정을 수행한 횟수: `int`

- '유효하지 않은' 의사 결정 목록:
  - 이동, 워프, 요청 게시 의사 결정의 추가 return**값** 또는 그 **형식**이 규칙에 맞지 않음
  - 이미 거점이 건설되어 있는 칸에서 거점 건설 행동을 선택
  - 이미 텔레포터가 건설되어 있는 칸에서 텔레포터 건설 행동을 선택
  - 거점이 없는 칸에서 수납 행동을 선택
  - 거점이 없는 칸에서 인출 행동을 선택
  - 텔레포터가 없는 칸에서 워프 행동을 선택하거나,
    텔레포터가 없는 칸으로 워프하도록 return**값**을 지정함
  - 이전에 게시했지만 아직 달성되지 않은 요청이 있는 시점에 요청 게시 행동을 선택, 또는 이미 거점/텔레포터 건설 요청이 게시되어 있는 칸에서 해당 요청을 중복 게시하도록 return**값**을 지정함
  - Core가 위 규칙들을 검사하는 과정에서 오류가 발생함(그럴 가능성은 희박할 듯)
- 아무튼 발생할 때마다 1씩 증가하며, 대기 의사 결정을 수행한 것으로 간주함.
- 게임 기본 설정상, 캐릭터가 유효하지 않은 의사 결정을 수행할 때마다 interactive에 관련 내용을 출력한 다음 사용자가 엔터 키를 칠 때까지 게임 진행을 중단하도록 되어 있음.
  - (대기하기 위해) 고의로 이상한 값을 return하는 것이 아니라면, 전반적인 효율을 높이기 위해 내 캐릭터 모듈의 전반적인 Data 흐름을, return문을 중심에 두고 검토해 보는 것을 권장함

0x0b – 의사 결정 도중 오류를 발생시킨 횟수: int

- NameError와 같이 평소에 자주 보던 오류가 발생하여 MakeDecision() 내용물 실행이 중단될 때마다 1씩 증가하며, 대기 의사 결정을 수행한 것으로 간주함.
- 내 캐릭터 **모듈**을 만들 때는 일단 F5를 눌러 interactive를 켜 둔 다음 IDLE의 자동 완성 기능을 적극적으로 활용해 가며 각 이름들을 적는 것을 추천함.
  - 코드 작성과 관련한 내용은 다음 설명서에서 더 자세히 나열할 예정
- 게임 기본 설정상, 캐릭터가 오류를 발생시킬 때마다 interactive에 관련 내용을 출력한 다음 사용자가 엔터 키를 칠 때까지 게임 진행을 중단하도록 되어 있음.
  - 오류가 몇 번째 줄 어디에서 발생했는지 확인하고, 어떻게 고칠 지 잘 모르겠다면 강사의 도움을 요청하는 것을 추천함

0x0c – 총 의사 결정 횟수: int

- 캐릭터 **모듈**의 MakeDecision()이 호출될 때마다 1씩 증가함. 따라서 이 **값**은 0x00 ~ 0x0b번째 **값**을 모두 더한 **값**과 같음.
- 전체 행동 횟수 대비 수집 행동을 몇 번 수행했는지를 체크하는 등의 용도로 활용 가능

0x0d – 의사 결정에 사용한 총 시간(현실 시간 기준, 초 단위): float

- MakeDecision() 호출 직전, 직후의 현재 시각을 재어 걸린 시간을 측정한 다음 이를 합산
- 여유가 있을 때 '동일한 의사 결정을 (현실 시간 기준) 더 빠르게' 수행하도록 **문장** 및 **수식**을 튜닝하고 싶은 친구들이 있을 것이므로 Core가 직접 재어 제공해 줌

0x0e, 0x0f – (여백)

0x10 – 수집한 총 자원량: int

- **(매우 중요)수집가 스타일 효율을 잴 때 가장 비중 있게 체크하는 항목임.**
- 수집 행동을 통해 자원을 집어 올릴 때마다 해당 자원량만큼 증가함.

0x11 – 이동 도착지와 중심 칸 사이의 거리의 합: int

- 이 **값**과 총 이동 횟수 **값**을 사용하여, 내 캐릭터가 어떤 영역에서 주로 활동했는지를 간접적으로 확인할 수 있음.
  - 중심 칸과의 거리에 따라 각종 자원량이 달라지므로, 능력치 조합을 점검하는 등의 용도로 활용할 수 있음(그렇긴 한데 튜토리얼 시나리오에서는 그리 중요하지 않을 듯)

0x12 – 점유에 사용한 총 자원량: int

- **(중요)수집가, 작업자 스타일 효율을 잴 때 체크하는 항목임.**
- 점유 행동을 통해 자원을 사용할 때마다 해당 자원량만큼 증가함.

0x13, 0x14 – 거점 건설, 텔레포터 건설에 사용한 총 자원량: int

- **(매우 중요)작업자 스타일 효율을 잴 때 가장 비중 있게 체크하는 항목임.**
- 각 행동을 통해 자원을 사용할 때마다 해당 자원량만큼 증가함.

0x15, 0x16 – 수납, 인출한 총 자원량: int

- 거점에 자원을 수납하거나 인출할 때마다 해당 자원량만큼 증가함.

0x17 – 워프한 칸 수: int

- 워프 행동을 수행할 때마다, 출발지와 도착지 사이의 거리만큼 증가함.
- 캐릭터의 전반적인 이동 효율을 체크하는 용도로 사용 가능.

0x18 – 자신이 게시했고 달성된 요청 수: int

- **(중요)작업자 스타일 효율을 잴 때 체크하는 항목임.**
- 자신이 게시한 요청이 달성될 때마다 1씩 증가.

0x19 – 대기를 통해 회복한 총 기력량: int

- 대기 행동을 수행할 때마다, 해당 행동을 통해 실제로 회복하는 기력량만큼 증가함.
  - 기력이 가득 찬 상태에서 대기 행동을 수행하면 0 증가
- 총 대기 횟수 **값**과 함께 활용하여 내 캐릭터의 전반적인 기력 회복 효율을 체크할 수 있음.

0x1a, 0x1b, 0x1c – 자신이 수집했고 점유, 거점 건설, 텔레포터 건설에 사용된 총 자원량: int

- **수집가 스타일 효율을 잴 때 추가로 체크하는 항목임.**
- 자신이 자원을 수집하고 직접 사용하거나, 수집한 다음 거점에 수납해 둔 자원을 자신 또는 다른 캐릭터가 인출하여 사용할 때마다, 해당 자원량만큼 증가함.
- 점수 산정에 사용되는 항목이지만, 거점 주변에서 수집 행동을 수행하도록 구성하는 경우 크게 신경 쓰지 않아도 무방함.
  - 새로운 거점을 고를 때, 다른 캐릭터들이 더 많은 곳이나 요청이 다수 게시되어 있는 곳을 우선적으로 고르도록 구성하면 이 항목 점수를 더 수월하게 올릴 수 있음

0x1d – 칸 점유에 기여하여 다른 캐릭터의 이동, 워프를 도와 얻은 점수: int

- **수집가, 작업자 스타일 효율을 잴 때 추가로 체크하는 항목임.**
- 캐릭터가 점유 행동을 수행하며 자원을 사용할 때마다, 그 칸에 대해 점유를 수행한 캐릭터와 그 자원을 수집한 캐릭터의 '기여 기록'을 남김(사용된 자원량만큼 누적).
- 해당 칸이 점유 완료된 이후, '다른 캐릭터'가 해당 칸에서 이동, 워프 행동을 수행하며 후딜레이 감소 효과를 받을 때마다 기여 기록에 따라 점수를 받음.
  - 내 캐릭터가 점유에 기여한 칸에서 이동, 워프해도 내 캐릭터의 점수는 오르지 않음
- 점수 산정에 사용되는 항목이지만 크게 고려하지 않아도 무방함.

0x1e – 거점 건설에 기여하여 다른 캐릭터의 수납, 인출, 대기를 도와 얻은 점수: int

- **수집가, 작업자 스타일 효율을 잴 때 추가로 체크하는 항목임.**
- 캐릭터가 거점 건설 행동을 수행하며 자원을 사용할 때마다, 그 칸에 대해 건설을 수행한 캐릭터와 그 자원을 수집한 캐릭터의 '기여 기록'을 남김(사용된 자원량만큼 누적).
- 거점 건설이 완료된 이후, '다른 캐릭터'가 해당 칸에서 하나 이상의 자원을 수납, 인출할 때, 또는 대기 행동을 수행할 때마다 기여 기록에 따라 점수를 받음.
  - 내 캐릭터가 기여한 칸에서 수납, 인출, 대기해도 내 캐릭터의 점수는 오르지 않음
- 점수 산정에 사용되는 항목이지만 크게 고려하지 않아도 무방함.

0x1f – 텔레포터 건설에 기여하여 다른 캐릭터의 워프를 도와 얻은 점수: int

- **수집가, 작업자 스타일 효율을 잴 때 추가로 체크하는 항목임.**
- 캐릭터가 텔레포터 건설 행동을 수행하며 자원을 사용할 때마다, 그 칸에 대해 건설을 수행한 캐릭터와 그 자원을 수집한 캐릭터의 '기여 기록'을 남김(사용된 자원량만큼 누적).
- 텔레포터 건설이 완료된 이후, '다른 캐릭터'가 해당 칸에서 다른 칸으로 워프할 때, 또는 다른 칸에서 해당 칸으로 워프할 때마다 기여 기록에 따라 점수를 받음.
  - 내 캐릭터가 기여한 칸에서 워프해도 내 캐릭터의 점수는 오르지 않음
  - 동일한 칸으로 워프할 때는 점수가 오르지 않음
- 점수 산정에 사용되는 항목이지만 크게 고려하지 않아도 무방함.

0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27, 0x28, 0x29
– 각 행동을 수행한 이후에 실제로 적용받은 후딜레이 합: float

- 각각 수집, 이동, 점유, 거점 건설, 텔레포터 건설, 수납, 인출, 워프, 요청 게시, 대기를 의미
  - **수식** 적을 때는 `0x20 + ret_gather` 와 같이 적을 수 있음
- '본 게임'은 게임 내부 시계 기준 10,000초 동안 진행되므로, 본 게임이 종료된 시점에 이 **값**들을 모두 더하면 10,000보다 조금 더 큰 **값**이 됨.
  - 내 캐릭터가 어떤 행동에 더 많은 시간을 사용했는지 체크하여 능력치 조합을 체크하는 용도로 사용 가능

0x2a, 0x2c – 점유 완료된 칸에서 이동, 워프한 횟수: int
0x2b, 0x2d – 점유 완료된 칸에서 이동, 워프할 때 절약한 후딜레이 합: float

- 후딜레이의 경우 기력 부족으로 증가하는 양은 고려하지 않음.
- 튜토리얼 시나리오 등에서 전반적인 이동 효율을 점검하기 위해 참고할 수 있음.

0x2e, 0x2f – (여백)

0x30, 0x31, 0x32 – 요청 달성을 위해 수집 및 사용한 총 자원량: int

- 각각 수집 요청, 거점 건설 요청, 텔레포터 건설 요청을 의미
- **수집가, 작업자 스타일 효율을 잴 때 추가로 체크하는 항목임.**
- 요청 달성을 위해 직접 수집, 거점 건설, 텔레포터 건설 행동을 수행할 때마다 직접 수집 및 사용한 자원량만큼 증가함.
- 이 항목은 과제 점수 산정 과정에서 보너스를 부여하기 위해 마련함: 현재 칸 주변에서 활동하지 않고 요청 달성에 기여하기 위해 이동하면서 발생하는 효율 감소를 보정하는 용도로 사용됨.
  - 따라서 요청 목록을 적극적으로 확인하면서 다른 캐릭터가 게시해 둔 요청을 달성하기 위해 활동하도록 만들어도 무방함

0x33, 0x36 – 자원을 들고 이동, 워프한 횟수: int
0x34, 0x37 – 이동, 워프할 때 들고 있던 자원량 합: int
0x35, 0x38 – 이동, 워프할 때 추가로 소모한 기력 합: int

- **(중요)내 캐릭터의 힘 능력치가 적절한지 여부를 확인하고 싶을 때 활용할 수 있는 항목임.**
- 총 이동/워프 횟수 **값**과 함께 사용하여 자원을 들지 않고 이동/워프한 횟수를 확인할 수 있음.
- 수집가 캐릭터를 만들 때, 자원량 및 기력 합과 횟수를 사용하여 평균을 잰 다음, 중심 칸과 현재 칸 사이의 거리에 따라 수집 및 대기 행동을 어떻게 수행할 것인지 조율해 둘 수 있음
  - 특히, 힘 능력치를 매우 높게 지정하는 경우 손을 가득 채우기 위해 수집을 추가로 수행하는 것보다 그대로 수납을 위해 이동하며 기력 소모량 감소 효과를 받는 것이 대기 횟수 절감 차원에서 더 유리할 수 있음

0x39 – 기력이 충분한 채로 점유, 거점 건설, 텔레포터 건설 행동을 수행한 횟수 합: int
0x3a, 0x3b – 기력이 충분한 채로 위 행동들을 수행하면서 적용받은, 절약한 후딜레이 합: float
0x3c – 기력이 충분한 채로, 빈 손으로 이동, 워프한 횟수 합: int
0x3d, 0x3e – 기력이 충분한 채, 빈 손으로 이동, 워프하며 적용받은, 절약한 후딜레이 합: float

- **(중요)내 캐릭터의 민첩성 능력치가 적절한지 확인하고 싶을 때 활용할 수 있는 항목임.**
- 이동, 워프 후딜레이의 경우 칸 점유로 인해 감소한 양은 고려하지 않음.
- 민첩성 능력치를 높일 때마다 후딜레이를 큰 폭으로 절약할 수 있으나, 다른 두 능력치를 줄임으로써 특정 작업 완료를 위해 요구되는 행동 횟수(대기 행동 포함)가 증가할 수 있음. 따라서, 특히 작업자 캐릭터를 만들 때는 이 항목의 값을 가운데에 두고 다른 항목들을 체크하면서 능력치 조합을 자신의 세부 스타일에 맞게 지정해 보는 것을 추천함.

0x3f – 행동 종료 시점의 현재 기력 합: int
0x40 – 기력 부족으로 후딜레이가 증가한 횟수: int
0x41 – 기력 부족으로 증가한 후딜레이 합: float

- **(중요)내 캐릭터의 체력 능력치가 적절한지 확인하고 싶을 때 활용할 수 있는 항목임.**
- 거점 주변에서 활동하는 수집가 캐릭터를 만드는 경우 이 항목들은 크게 고려하지 않아도 됨.
- 작업자 캐릭터를 만들 때는, 점유 및 건설 행동의 기력 소모량이 매우 크게 지정되어 있으므로 행동 종료 시점의 기력 평균을 재어 전반적인 기력 관리 상황을 체크해 보는 것을 권장함.
  - 행동 '종료' 시점에 측정하므로 합 및 평균은 최대 기력량 대비 매우 낮을 수 있음
  - 의사 결정을 수행하는(행동을 시작하는) 시점에 현재 기력이 0 이상이면 후딜레이 패널티를 받지 않음. 따라서 체력 능력치가 낮더라도 패널티 없이 점유 및 건설 행동을 수행할 수 있으며, 다른 두 능력치를 높여 자원 운반 능력을 높이거나 후딜레이 절약 효과를 받는 것이 효율 측면에서 더 유리할 수 있음
- 특별한 이유가 없다면 기력이 없는 상태에서 대기 이외의 행동을 선택하지 않는 것을 권장함.

부록#3: 각종 **이름**들과 **수식**들

이 단원에는 게임 내에서 사용되는 각종 Data 및 해당 **값**을 **계산**하기 위해 사용되는 **수식**들이 나열되어 있습니다. 내 캐릭터 **모듈**을 만들 때 직접 적을 수 있는 **이름**은 myPlane과 같은 일반 글자로, 여러분이 유효하게 적을 수 없을 **이름**은 *Score_View*와 같이 기울임체로 적어 두었습니다. 또한, 가독성을 위해 infos **이름 사전**에 등재된 **이름**들은 풀 네임 없이 그냥 그 **이름**을 적어 두었습니다. 코드 작성 도중 궁금한 점이 있다면 Ctrl + F 등의 기능을 사용하여 해당 **이름**을 검색해 보면 좋겠습니다.

내 캐릭터 관련, 게임 진행 도중 고정되어 있는 Data

- `idx_me: int`
  - 캐릭터마다 고유한 **index 값**
  - 이름이 더 '빠른' 캐릭터에 대해 더 작은 **값**이 지정됨
    (최종 버전에서는 제출한 순서대로 지정할 예정)
  - 캐릭터 위치 목록, 캐릭터별 최근 행동 목록, 요청 목록 등을 조회할 때 활용 가능
- `max_r_carrying: int = stats[1]`
  - 캐릭터가 들 수 있는 최대 자원량(힘 능력치 **값**과 동일)
- `coef_stamina_cost_reduction_after_move: int = stats[1] // 2`
  - 자원을 들고 이동, 워프할 때의 기력 소모량 감소 계수
- `coef_delay_after_move: float = 0.9 ** stats[2]`
  - 자원을 들지 않고 이동, 워프할 때의 후딜레이 적용 계수
- `coef_delay_after_work: float = 0.75 ** stats[2]`
  - 자원을 들고 점유, 거점 건설, 텔레포터 건설할 때의 후딜레이 적용 계수
- `max_stamina: int = stats[3] * 4`
  - 최대 기력량
- `coef_stamina_heal: int = stats[3]`
  `coef_stamina_heal_on_base: int = stats[3] * 2`
  - 거점이 없는, 있는 칸에서 대기 행동을 한 번 수행할 때 회복되는 기력량
- `max_stamina: int = stats[3] * 4`
  - 최대 기력량

내 캐릭터 관련, 게임 진행 도중 갱신되는 Data

- `pos_me: tuple`
  - 현재 위치 좌표
- `recent_action_me: tuple`
  - 마지막에 수행한 행동 정보
  - 행동 정보 `tuple`의 `0`번째 칸에는 해당 행동에 대한 숫자(예: 수집이면 `0`)가 담겨 있으며, 이동, 워프, 요청 게시 행동의 경우 1, 2번째 칸에 그에 대한 추가 Data가 담겨 있음
- `r_carrying: int`
  - 들고 있는 자원량
- `stamina: int`
  - 현재 기력
- `myScore: *Score_View*`
  - 점수 목록(세부 내용은 부록#2 참조)
  - `[ ]` **수식**을 적어 각 점수 항목을 조회하도록 구성할 수 있음
  - 게임 종료 이후에 결과 확인을 위해 사용하는 것을 권장함
- `myRequest: None 또는 tuple`
  - 내 캐릭터가 최근에 게시한 요청 정보. 이전에 요청을 게시한 적이 없다면 `None`
  - 요청 정보 `tuple` 내용:
    `[0]` – 요청마다 고유한 일련번호: `int`
    `[1]` – 요청을 게시한 좌표: `tuple`
    `[2]` – 요청 종류: `int`
      0 – 자원 수집
      1 – 거점 건설
      2 – 텔레포터 건설
    `[3]` – 요청 달성에 필요한 자원량: `int`
  - 이전에 요청을 게시한 적이 있고, 내 최근 요청이 이미 달성되었는지 여부를 확인하고 싶을 때는 **수식** `infos.request_isCompleted(infos.myRequest)` 를 사용하면 됨
  - 이전에 요청을 게시한 적 있고, 내 최근 요청을 달성하기 위해 남은 자원량을 확인하고 싶을 때는 **수식** `infos.request_getAmount`어쩌구`(infos.myRequest)` 를 사용하면 됨

각종 목록 Data (평면 관련 정보 제외)

여기에 나열된 Data는 [ ] **수식**, for문, `len()` 호출식 등을 적어 가며 내용물을 조회할 수 있음.

- `pos_characters:` *Pos_Characters_View*
  - 각 캐릭터들의 현재 위치 좌표 `tuple`들이 **index** 순서대로 담겨 있음

- `recent_action_characters:` *Recent_Action_Characters_View*
  - 각 캐릭터들의 마지막 행동 정보 `tuple`들이 **index** 순서대로 담겨 있음
  - 행동 정보 `tuple`의 0번째 칸에는 해당 행동에 대한 숫자(예: 수집이면 `0`)가 담겨 있으며, 이동, 워프, 요청 게시 행동의 경우 1, 2번째 칸에 그에 대한 추가 Data가 담겨 있음

- `pos_bases:` *Pos_Bases_View*
  `pos_teleporters:` *Pos_Teleporters_View*
  - 모든 거점, 텔레포터의 위치 좌표 `tuple`들이 건설 완료된 순서대로 담겨 있음

- `requests_active:` *Requests_Active_View*
  `requests_all:` *Requests_All_View*
  - 달성되지 않은, 모든 요청 정보 `tuple`들이 게시된 순서대로 담겨 있음
  - 요청 정보 `tuple` 내용:
    `[0]` – 요청마다 고유한 일련번호: `int`
    `[1]` – 요청을 게시한 좌표: `tuple`
    `[2]` – 요청 종류: `int`
      0 – 자원 수집
      1 – 거점 건설
      2 – 텔레포터 건설
    `[3]` – 요청 달성에 필요한 자원량: `int`
  - 어떤 요청 `rq` 에 대해 해당 요청을 달성하기 위해 남은 자원량을 확인하고 싶을 때는 **수식** `infos.request_getAmount어쩌구(rq)` 를 사용하면 됨

기타 Data (평면 관련 정보 제외)

- `current_time: int`
  `end_time: int`
  - 게임 내 시계 기준 현재 시각과 종료 예정 시각

- `numberOfCharacters: int`
  - 게임에 참여중인 캐릭터 수(`len(pos_characters)` 와 동일)

평면 관련 Data

[ ] **수식**을 사용하여 평면 위 특정 칸에 대한 Data를 조회할 수 있음. 단, 각 평면의 크기는 게임 진행 도중 계속 확장되므로 평면의 길이를 재거나 평면 위 모든 칸에 대해 작업을 수행하는 **반복** 흐름을 구성하는 것은 권장하지 않음.

- `myPlane:` *MyPlane_View*
  - 내 캐릭터의 개인 평면을 의미
  - [ ] **수식**을 사용하여 특정 칸에 놓여 있는 자원의 양(`int` **형식**)을 조회할 수 있음
    (각 칸에 처음 놓여 있는 자원량 공식은 후술)
- `publicPlane:` *PublicPlane_View*
  - 공유 평면을 의미
  - [ ] **수식**을 사용하여 특정 칸에 대한 정보를 조회할 수 있음
- 공유 평면 위 칸 정보(*PublicCell_View* **형식**) 내용:
  - `time_read: int = current_time`
    - 게임 내 시계 기준, 이 칸 정보 Data를 조회한([ ] **수식**을 **계산**한) 시각
  - `r_stored: int`
    - 거점에 수납되어 있는 자원량(거점이 없는 경우 `0`)
  - `r_toOccupy: int`
    `r_toBuildBase: int`
    `r_toBuildTeleporter: int`
    - 점유, 거점 건설, 텔레포터 건설 완료까지 필요한 자원량(`0`인 경우 완료)
      (각 칸에 대한 필요 자원량 공식은 후술)
  - `count_characters: int`
    - 해당 칸에 있는 캐릭터 수
  - `coef_delay_after_move: int`
    - 해당 칸에서 이동, 워프할 때의 후딜레이 계수
- 주의: 공유 평면 위 칸 정보는 매 번 [ ] **수식**을 **계산**할 때마다 그 사본을 만들어 다루게 됨.
  - 현재 시점의 칸 정보를 담아 두었다가 다음 의사 결정 도중에 가져와 사용할 수 있음
  - 반면 [ ] **수식**을 너무 자주 **계산**하는 경우 컴퓨터에 부담을 줄 수 있으므로, `input()` 호출식을 다룰 때와 비슷한 느낌으로, 일단 한 번 **계산**해서 어딘가에 담아 둔 다음 천천히 꺼내 가며 사용하는 것을 권장함

마지막, 자원량 관련 수식들

- 개인 평면 위 특정 칸 (x, y)에 놓여 있는 자원량
  - `5 + ((abs(x) + abs(y)) // 32)`
- 공유 평면 위 특정 칸 (x, y)을 점유하기 위해 필요한 자원량
  - `int(25 * 1.6 ** ((abs(x) + abs(y))/512))`
- 공유 평면 위 특정 칸 (x, y)에 거점을 건설하기 위해 필요한 자원량
  - `int(50 * 6.0 ** ((abs(x) + abs(y))/512))`
- 공유 평면 위 특정 칸 (x, y)을 텔레포터를 건설하기 위해 필요한 자원량
  - `int(100 * 1.5 ** ((abs(x) + abs(y))/512))`
- 중심 칸과의 거리에 따른 자원량 예시 표:

거리	놓인 자원량	점유 필요량	거점 필요량	텔포 필요량
0	5	25 (5칸)	50 (10칸)	100 (20칸)
32	6	25 (5칸)	55 (10칸)	102 (17칸)
64	7	26 (4칸)	62 (9칸)	105 (15칸)
128	9	28 (4칸)	78 (9칸)	110 (12칸)
256	13	31 (3칸)	122 (10칸)	122 (10칸)
384	17	35 (3칸)	191 (12칸)	135 (8칸)
512	21	40 (2칸)	300 (15칸)	150 (8칸)
768	29	50 (2칸)	734 (26칸)	183 (7칸)
1024	37	64 (2칸)	1800 (49칸)	225 (7칸)

  - 필요량 항목 괄호 안에 주변 몇 칸에서 수집을 진행해야 충당할 수 있는지 기록해 둠

- 요약:
  - 개인 평면에 놓여 있는 자원량은 기본 5에 거리 32마다 1씩 linear하게 증가하며,
    점유 및 건설에 필요한 자원량은 거리 대비 exponential하게 증가함
  - 거리 256 부근에서 거점 건설에 필요한 자원량이 텔레포터 필요 자원량을 추월함.
    따라서 해당 범위 바깥에서는 새 거점을 건설하기 어려울 수 있음(다음 장 그래프 참조)
  - 반면 점유에 필요한 자원량은 거리에 따른 증가율이 적은 편이며, '본 게임'에서 상정하는 현실적인 범위 안에서는 중심 칸에서 멀리 있을수록 칸당 수집량 대비 점유 필요량 비율이 줄어든다 볼 수 있음(다음 장에 있는 그래프 참조)
  - 이러한 점을 감안하여, 일단 기본적인 작업들을 수행하기 위한 Code 및 Data 흐름을 구성해 둔 다음, 능력치 조합을 바꾸어 가며 내 캐릭터가 평면 위 어떤 영역에서 주로 활동할 것인지 결정하고, 제출 전까지 세부적인 튜닝을 진행해 보는 것을 권장함

- 중심 칸과의 거리에 따른 자원량 예시 그래프: