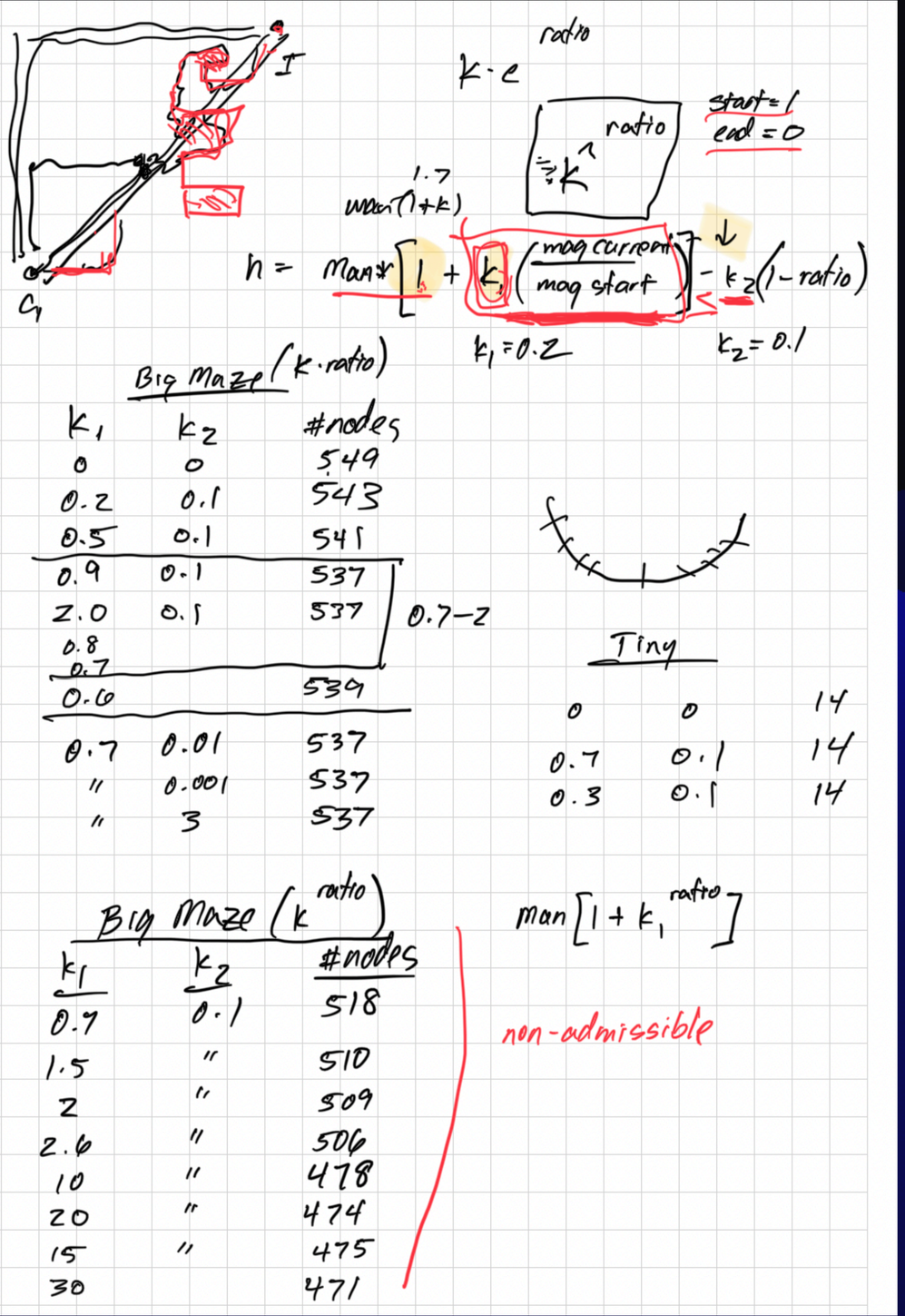
PACMAN FINAL PROJECT
“Average Joes”

LT Alex Huang, USN
LT Cam Mundy, USN
CDR Donny Peltier, USN

CS3310, Naval Postgraduate School, Fall 2022

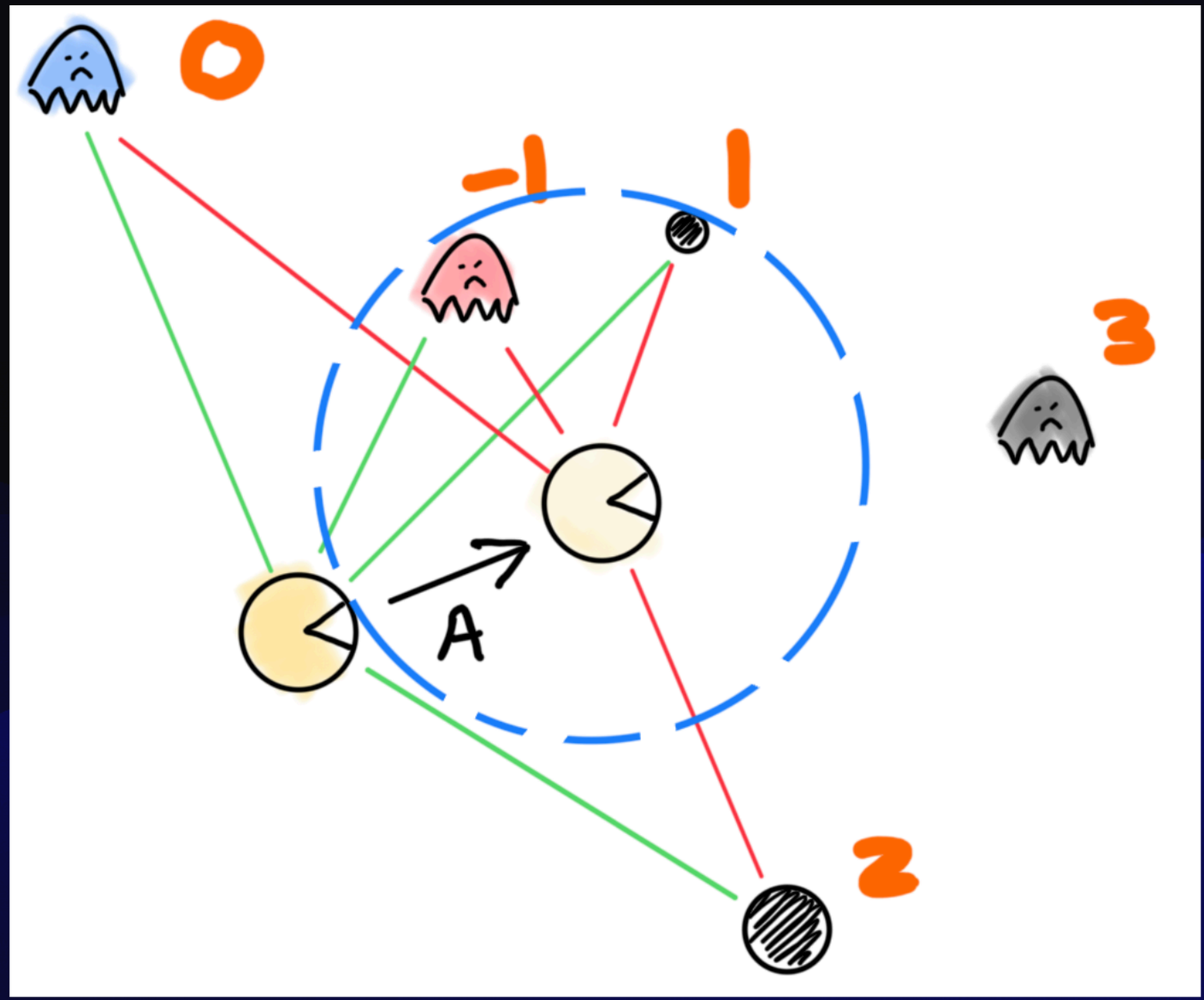
A* Heuristic

- Closer approximation = better
  - Manhattan > Euclidean
- Traffic analogy
  - Fade out correction k1 -> goal
- Ensure admissible
  - Fade in -k2 -> goal



Reflex Agent

- Current state vs. new state




```
return (successorGameState.getScore()+foodscore+ghostscore+capsulescore)
```

Minimax Implementation

- Implemented the minimax algorithm discussed in class, with selected action
- Dispatch function only needs to kick off the recursive calls to the terminal nodes or max depth
- Tracks depth by passing to next layers of recursion instead of using a global
- Differs slightly from textbook algorithm, which only passes value and leaves action selection to the dispatch function

```
function MINIMAX-SEARCH(game, state) returns an action
    player ← game.TO-MOVE(state)
    value, move ← MAX-VALUE(game, state)
    return move

function MAX-VALUE(game, state) returns a (utility, move) pair
    if game.IS-TERMINAL(state) then return game.UTILITY(state, player), null
    v ← −∞
    for each a in game.ACTIONS(state) do
        v2, a2 ← MIN-VALUE(game, game.RESULT(state, a))
        if v2 > v then
            v, move ← v2, a
    return v, move

function MIN-VALUE(game, state) returns a (utility, move) pair
    if game.IS-TERMINAL(state) then return game.UTILITY(state, player), null
    v ← +∞
    for each a in game.ACTIONS(state) do
        v2, a2 ← MAX-VALUE(game, game.RESULT(state, a))
        if v2 < v then
            v, move ← v2, a
    return v, move
```

Alpha-Beta Pruning Implementation

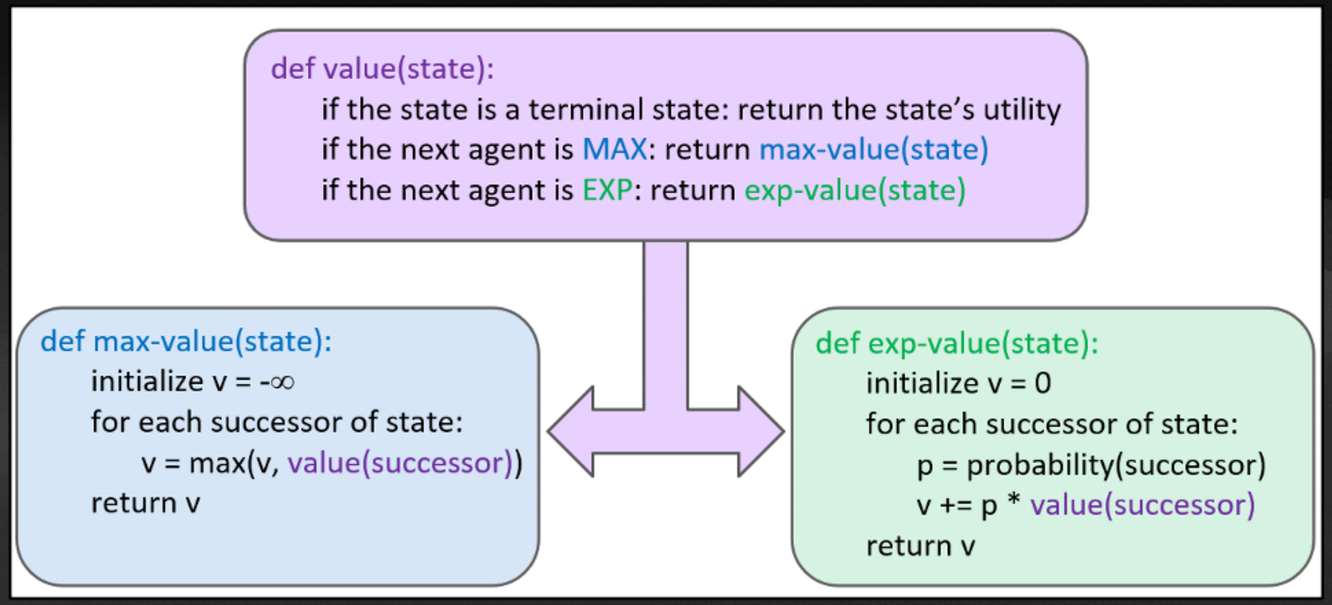
- Expanded Minimax algorithm to track and compare alpha/beta values to root
- Updated alpha/beta values passed between layers of recursion

```
function ALPHA-BETA-SEARCH(state) returns an action
  v ← MAX-VALUE(state, −∞, +∞)
  return the action in ACTIONS(state) with value v

function MAX-VALUE(state, α, β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for each a in ACTIONS(state) do
    v ← MAX(v, MIN-VALUE(RESULT(s,a), α, β))
    if v ≥ β then return v
    α ← MAX(α, v)
  return v

function MIN-VALUE(state, α, β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← +∞
  for each a in ACTIONS(state) do
    v ← MIN(v, MAX-VALUE(RESULT(s,a) , α, β))
    if v ≤ α then return v
    β ← MIN(β, v)
  return v
```

Figure 5.7 The alpha–beta search algorithm. Notice that these routines are the same as the MINIMAX functions in Figure 5.3, except for the two lines in each of MIN-VALUE and MAX-VALUE that maintain α and β (and the bookkeeping to pass these parameters along).

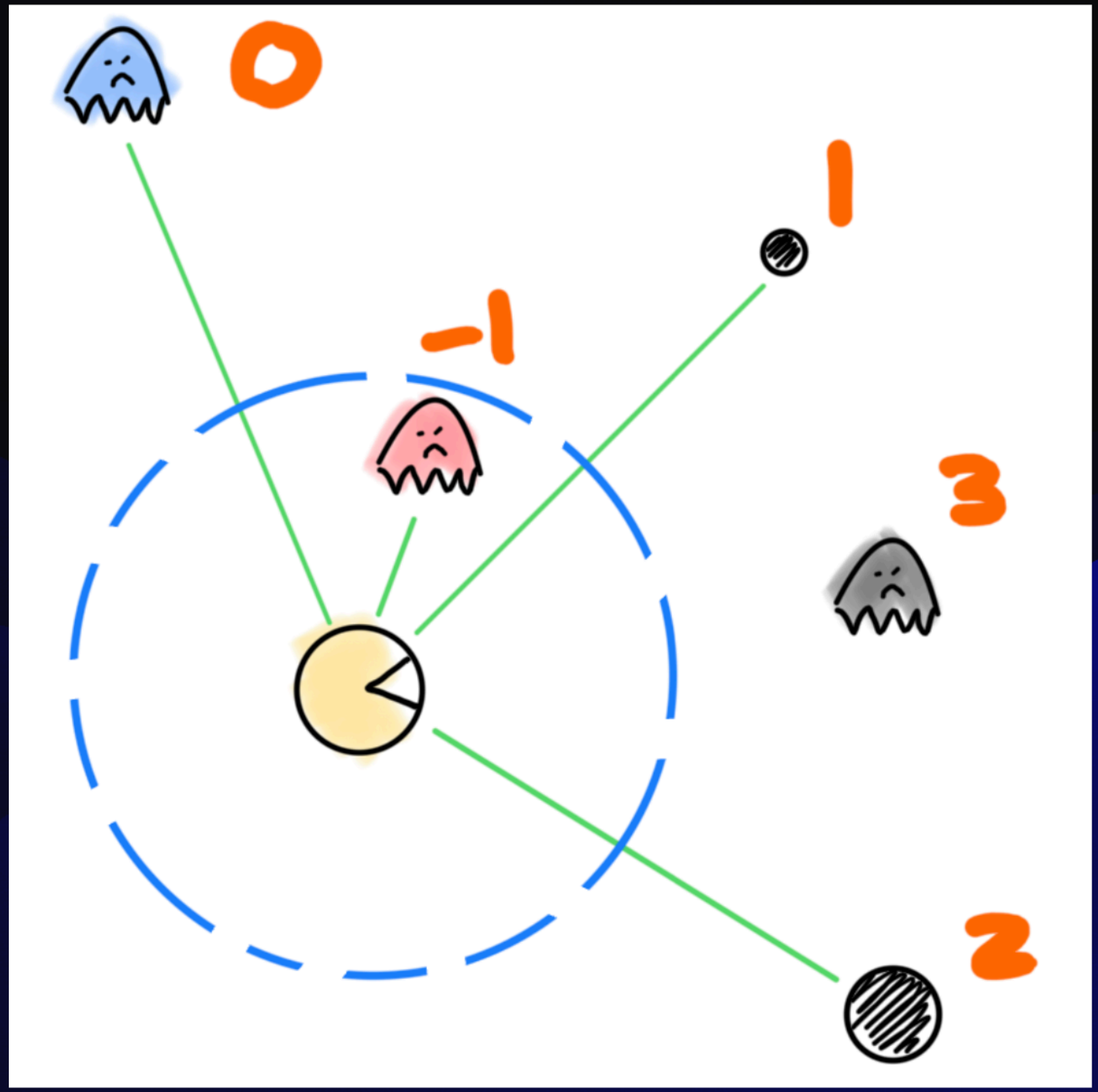
Expectimax Implementation

- Replaced min (ghost) players with chance players to model "random" ghost movement instead of "minimized" or optimal movement
- maxValue calls expValue instead of minValue
- All legal actions have equal probability



Evaluation Function

- Current state only




```
return (successorGameState.getScore()+foodscore+ghostscore+capsulescore)
```

Lessons Learned

- Pseudocode from course lectures were easily adaptable to Pacman and real-world implementation
- Team fell victim to basic programming errors
- Collaboration tools are key to programming as a team
- Successfully used NPS GitLab for version control and sharing progress
- Common cross-platform baseline development environment in VSCode for all team members—no issues between different operating systems
- Foundational understanding of recursion was important for Minimax and Expectimax implementation