EECS 183: Elevators

Hailey Brady, Pranav Varshney, Daniel Heilman, Bianca Brie

STRATEGY

Our strategy was a 3 step process. First we checked what move to make ie. pass, pickup, service. If pickup, we chose direction where most people are going ie. up or down. If service, we sent the elevator to the floor with the most people.

```cpp
/*
 * Copyright 2023 University of Michigan EECS183
 *
 * Building.cpp
 * Project UID 28eb18c2c1ce490aada441e65559efdd
 *
 * Pranav Varshney, Hailey Brady, Bianca Brie, Daniel Heilman
 * heilmand, hbrady, pvarsh, briebi
 *
 * Final Project - Elevators
 */
#include "AI.h"
#include <cassert>
// This file is used only in the Reach, not the Core.
// You do not need to make any changes to this file for the Core
string getAIMoveString(const BuildingState& buildingState) {

    string getAIMove = "";
    // pass move

    // checks if elevator is servicing or not
    bool isServicingArray[NUM_ELEVATORS];
    for (int i = 0; i < NUM_ELEVATORS; i++) {
        isServicingArray[i] = buildingState.elevators[i].isServicing;
    }

    // checks to see if current floor has people or not
    int currFloor[NUM_ELEVATORS] = {0};
    int numPeople;
    bool isPeople[NUM_ELEVATORS] = {0};
    for (int i = 0; i < NUM_ELEVATORS; i++) {
        currFloor[i] = buildingState.elevators[i].currentFloor;
        numPeople = buildingState.floors[currFloor[i]].numPeople;

        if (numPeople != 0) {
            isPeople[i] = true;
        }

        else {
            isPeople[i] = false;
        }
    }

    //stores information about each floor
    int floorPeople[NUM_FLOORS];
    bool upRequest[NUM_FLOORS];
    bool downRequest[NUM_FLOORS];
    for (int i = 0; i < NUM_FLOORS; i++) {
        floorPeople[i] = buildingState.floors[i].numPeople;
        upRequest[i] = buildingState.floors[i].hasUpRequest;
        downRequest[i] = buildingState.floors[i].hasDownRequest;
    }

    //Find index of floor with most people & set floorToPickup to that value
    int floorToPickup = 0;
    int temp = 0;
    int temp2 = 0;
    int count = 0;
    for(int i = 0; i < NUM_FLOORS; i++) {
        temp = floorPeople[i];
        for(int j = 0; j < NUM_FLOORS; j++) {
            temp2 = floorPeople[j];
            if(temp < temp2) {
                count++;
            }
        }
        if(count == 0) {
```

IMPLEMENTATION

First, we checked if all elevators were servicing through a loop. We implemented our strategy by creating a loop in order to store the floor with the most people as a count variable and setting floorToPickup to this value. We used this to check for the floor with the most people, and proceeded further with that information

```cpp
        if(count == 0) {
            floorToPickup = i;
        }
        count = 0;
    }



    // writes AI move for pickup and service move
    int elevatorInt = 0;
    char elevatorChar = ' ';
    string elevatorString = "";

    int floorInt = 0;
    char floorChar = ' ';
    string floorString = "";

    //steps through each elevator so set move
    for(int i = 0; i < NUM_ELEVATORS; i++) {
        //first makes sure the elevator isn't already servicing
        if(!isServicingArray[i]) {
            //if there are people on the floor, return pickup move
            if(isPeople[i]) {
                elevatorInt = i;
                elevatorChar = elevatorInt + '0';
                elevatorString = elevatorChar;

                getAIMove = "e" + elevatorString + "p";
                return getAIMove;

            //if there aren't people, sets service move and moves on to the next
            //elevators
            } else if(!isPeople[i]) {
                elevatorInt = i;
                elevatorChar = elevatorInt + '0';
                elevatorString = elevatorChar;

                if(floorToPickup == currFloor[i]) {
                    floorToPickup--;
                }

                floorInt = floorToPickup;
                floorChar = floorInt + '0';
                floorString = floorChar;
                getAIMove  = "e" + elevatorString + "f" + floorString;
            }
        }
    }

    return getAIMove;
}
string getAIPickupList(const Move& move, const BuildingState& buildingState,
                       const Floor& floorToPickup) {
```

```cpp
    string getAIPickupList = "";


    int numPeople = floorToPickup.getNumPeople();
    int peopleUp[MAX_PEOPLE_PER_FLOOR] = {0};
    int peopleDown[MAX_PEOPLE_PER_FLOOR] = {0};
    int upRequestInt = 0;
    int downRequestInt = 0;
    int diffFloors = 0;

    //stores data on the number and indexes of people going up and down
    for (int i = 0; i < numPeople; i++) {
        diffFloors = floorToPickup.getPersonByIndex(i).getTargetFloor() - floorToPickup.getPersonByInde

        if (diffFloors > 0) {
            peopleUp[upRequestInt] = i;
            upRequestInt ++;
        }

        else {
            peopleDown[downRequestInt] = i;
            downRequestInt ++;
        }
    }


    int peopleInt = 0;
    char peopleChar = ' ';
    string peopleString = "";

    //if more people are going up than down, sets up move, if not sets down move
    if (upRequestInt > downRequestInt) {
        //sets pickup list for people going up
        for (int i = 0; i < upRequestInt; i++) {
            peopleInt = peopleUp[i];
            peopleChar = peopleInt + '0';
            peopleString = peopleChar;

            getAIPickupList += peopleString;
        }
    }

    //sets pickup list for people going down
    else {
        for (int i = 0; i < downRequestInt; i++) {
            peopleInt = peopleDown[i];
            peopleChar = peopleInt + '0';
            peopleString = peopleChar;

            getAIPickupList += peopleString;
        }
    }

    return getAIPickupList;
}
```

EVALUATION

Our code did a good job in maxmizing points, picking up at floors with the most people. However, our code did not consider the optimization of the movement of the elevator nor explosion of people.