CEC 2019
CCI 2019

PROGRAMMING/ PROGRAMMATION

Rulebook/ Livre de règles

Thomas Dedinsky
programming@cec.cfes.ca

Programming

General Rules

The goal of the programming category is to encourage engineering students to produce a piece of readable software. The teams will use their software development skills, their technical writing abilities, and their project management skills to design a solution to a posed problem. This solution will then be presented to a panel judges and assessed by them. The winning solution will not necessarily be to the most technically correct but the one that has the most real world application and is most thoroughly thought out.

The Programming team will be comprised of a maximum of four competitors. At least half of the design team must be representing an accredited engineering program at an active CFES-member school. The presentation and any presentation materials can be done in either English or French, but must be consistent in language.

The topic will be a real life problem found in any professional industry which can be solved through the application of programming. The type of industries can include, but are not limited to, finance, health, transportation, manufacturing and construction. Although not completely needed, a team that is formed of students from more than one engineering discipline is advised, as it would help to develop a complete solution.

Competition Schedule

At least seven days (168 hours) prior to the competition, the main theme(s) of the competition will be announced to the competitors, judges and public. Exact specifications of equipment available to teams during the competition (computers storage devices, available programs, etc.) will also be announced at this time. Each team member will electronically be given access to a package containing this information, and it is the competitor's responsibility to ensure that they have received the package.

The problem must be presented to all competitors and judges at the beginning of the competition during the briefing period. The competition director must provide detailed explanations of what is expected from the competitors, both orally and in writing. Specifically, a copy of this package outlining the problem definition, design and presentation requirements, rules, marking scheme and any other information deemed necessary by the competition director on the day of the competition will be provided via a link to an online repository.

Only the competition director is allowed to answer questions during the briefing period. Immediately following the briefing, for at least 15 minutes before the design phase starts, until one (1) hour before the design phase ends, competitors will have access to a form for which they can submit questions to the competition director. Once the design phase starts, questions answered by the competition director must only be related to the deliverable content. Responses to valid questions to the competition director must be made available to all of the competitors within 15 minutes of the question being asked. A copy of the responses must be provided to the judges prior to the presentations.

Teams will be given 5-8 hours (at the discretion of the organizers, length to be given to competitors at least seven days in advance) to develop their solutions, produce all required deliverables, and prepare their presentations. All deliverables shall be

submitted prior to the end of the provided time. This must include the code which will be used to evaluate the team's solution and their presentation. Only the code provided to the competition director will be used for evaluation. Competitors must be allowed a minimum of one hour to rest before the presentation phase starts.

Competitors will have a maximum of 20 minutes to present their solutions. All team members must be present and participate in the presentation or be penalized by the judges. Judges then have a maximum of 10 minutes to ask questions. Judges can ask a question at anytime during the presentation. The clock should be stopped during these interruptions.

The Official Timekeeper must be responsible for enforcing time limits during the question period, solution development and presentations, as stated in the rules. The following rules must be adhered to by the official timekeeper. During the design phase, time is started when the teams arrive at their respective workstations. The time remaining in the competition must be announced 2 hours, 1 hour, 30 minutes, 15 minutes and 5 minutes before the deadline through the CEC 2019 Slack.

Background

As automation becomes increasingly present in our lives, blue-collar jobs with simple forms of labour are now being replaced with robots. This allows the company to complete the same amount of work as previously at a fraction of the cost. Assembly line work is the first being automated, but it is slowly expanding into other sectors such as services. Companies who would like to save money and be able to grow more rapidly are already turning to these solutions. By 2020, McDonalds will have most of its 14,000 locations and hundreds of thousands of staffed cashiers replaced by order kiosk. Most warehouse moving jobs, especially at bigger companies like Amazon, have been replaced with robots. With automation increasing in other industries, such as transportation and home appliances, it's inevitable that the only part of the services industry of the future that will be human are the customers.

Res-TRON-t Solutions (RS) is an automation company that specializes in creating robots that serve purposes in restaurant settings. They've become very successful due to their production line cleaning service product for large food courts and drive-thru automation product line, however they would like to grow their market share in the semi-formal eatery industry. Establishments that serve food but also are used for other purposes, such as board game cafes or Starbucks, usually leave their staff to clean up their mess, which usually involves transferring garbage, recyclables and organics to their disposal bins. These establishments usually have the closing staff clean up the place after hours, as well as have them check tables and clean periodically throughout the day. Additionally, waste will often not be properly sorted which causes lots of recyclables and organics to be improperly sent to landfills, and garbage to be sent to recycling or organics facilities.

However, RS would like to reduce the need of staff even further and increase the sorting effectiveness of trash. Due to their licensing of other companies' technology, engineers at the company have been able to program a device using a modified autonomous vacuum cleaner that can interact with surfaces off the floor, such as tables

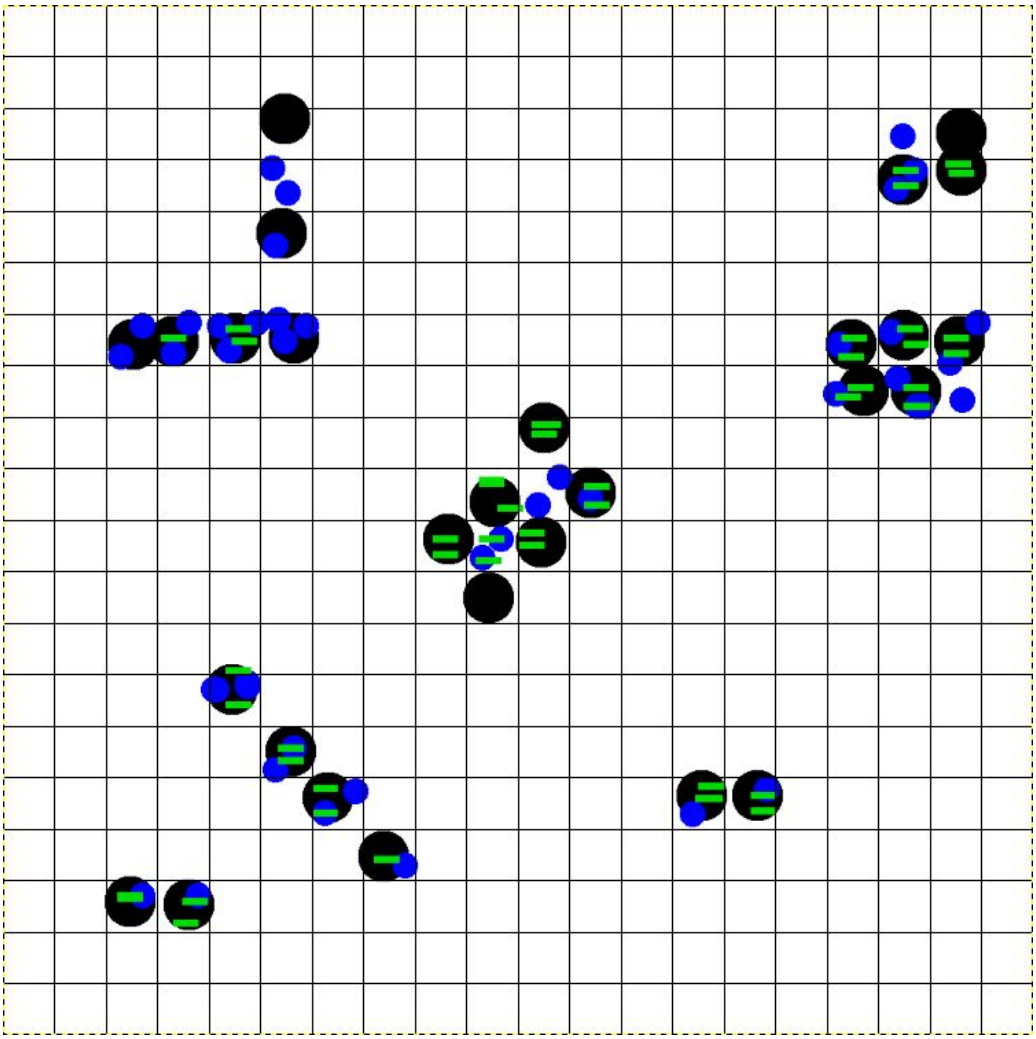
and bar stools. It can collect pieces of trash, and place them in the appropriate disposal bins.

RS would like you to develop a software solution to help control the robot and enable it to clean the trash efficiently. Garbage, recyclables, and organics will be scattered in a dining room and it is your job to navigate the robot to clean all of them. This software will communicate with the robot via REST API, and will be given access to various commands that will help it locate, retrieve, and place tableware. Efficiency is key, and software that can help optimize this problem will help it be used in real-time situations. You must present your product to the company board (judges) by focusing on its simplicity, ingenuity, and ability to achieve its desired outcome. The code for your application must be submitted using email and optionally a USB device, along with instructions on how to run it in a README file. This software can be written in any language and will communicate using rate-limited API calls to a simulation server.

The Challenge and Design Requirements

You will be communicating with a specialized autonomous cleaner to carry out all phases of cleaning. You will help control the robot and enable it to clean the trash effectively. Various garbage, recyclables, and organics will be scattered on multi-level surfaces such as tables and bar stools and it is your job to navigate the robot to clean all of them. You will basically be engineering an artificial intelligence for this device.

Sample eatery space layout. Black: Garbage, Blue: Recycle, Green: Organic. Note that this image only indicates what items would be visible in a first scan, not including overlaps, and only accurately represents the recyclables and organics count.



This software will communicate with the robot via REST API, and will be given access to various commands that will help it locate, retrieve, and place trash. Efficiency is key, and software that can help optimize this problem will help it be used in real-time situations.

Objectives

Your goal with this task is to design a robot that can cleanup the entire eatery space in the fastest possible amount of time.

You must create software to accomplish this task. This software will be completely autonomous, running completely without additional input until all of the trash has been cleaned up, aiming to complete the task in the least amount of time needed. Your time will not be tested in real-time, instead using the absolute time units provided in the corresponding instance. During development, you may test your solution with the development instance as many times as you want.

You must present your product to the company board (judges) by focusing on its simplicity, ingenuity, and ability to achieve its desired outcome. The code for your application must be submitted using email and optionally a USB device, along with instructions on how to run it in a README file, and your presentation. This software can be written in any language and will communicate using rate-limited API calls to a simulation server.

API

All communication with the robot will be done using the robot's API, specifically using RESTful API. Given this, your code can be written with any language that is capable of communicating with the server using HTTP requests, with some examples being listed in the Rules section. Responses will be in JSON format, with the structure of the

response being the updated instance if it's a success, or an error message if it's a failure.

The response will have a variable {type}. If {type:"ERROR"}, it will have an {error} and {message} variable with additional information. An error will be an invalid request that should never work, like using an invalid token. If {type:"FAILURE"}, it will have an {failure} and {message} variable with additional information. A failure will be an action you cannot currently do, like move past a wall. If {type:"SUCCESS"}, it will have a {payload} variable. The top level variables in the payload will all be useful information for the programming challenge that you can use to get feedback on your actions. Examples of the variables can be seen in the documentation section. You will also have constants that tell you useful information such as the room dimensions or what the time costs are for an action.

Methods

In order to use the API, you must have a token, given to you by the programming competition lead before the competition begins. Use this in the Header with the key "token". The following methods will all be POST, and will begin with the url http://cec2019.ca

- /instance - Creates an instance for the programming competition. You can also use GET and DELETE respectively with this method.
- /finish - Use this method to declare that you are finished collecting all of the items in the room.
- /turn/:Direction - Turns the cleaning robot in a cardinal direction specified by a single character. N is +ve y-axis, S is -ve y-axis, E is +ve x-axis, W is -ve x-axis.
- /move - Moves the robot one block in the direction it is facing.
- /scanArea - Scans for nearby trash in a diamond shape that is within a specified radius.

- /collectItem/:ID - Collects an item previously scanned. Note that you must be on the block of the item in order to collect the item.
- /unloadItem/:ID - Unloads the item when the robot is at the the proper disposal bin.

Restrictions

- Be aware that the capacity of the disposal bins is much less than the amount of trash in the eatery, and takes a long time to run. This limit does not apply to the robot. Plan wisely.
- Items may be hidden under each other, and the robot isn't able to recognize how certain objects overlap. You will need to check to make sure you aren't missing something after collecting an item. Two items may be at the same location but don't have to overlap.
- Distributions of items tend to mirror a real-world eatery, so try not to make too many assumptions about biases that don't exist naturally.
- In order to prevent DDoS attacks or overusage, responses from api calls are delayed by about half a second. You shouldn't try simultaneous api calls at any time. Intentionally DDoSing the server or other hacking measures will result in immediate disqualification.
- Constants are constant to the instance; they may change between the testing instance and the presentation instance.

Materials

An access token to the simulation server via API will be provided. A design room with at least one table, four chairs, a whiteboard or blackboard, paper and pencils/pens for writing, as well as access to a computer with internet connectivity, will be provided during the design phase. A presentation room with a digital projector, a computer containing the team's presentation file, a whiteboard or blackboard, and simultaneous translation equipment if required will be provided during the presentation phase.

Each competitor is allowed to bring any reference material, such as background research or course notes, as well as a computer and/or peripheral storage device with any electronic material stored onto it. If you are bring electronic material on a peripheral without your own computer, please check with the competition director to make sure the format of your electronic information will be accessible using the computer provided.

It is expected that the teams participating in this competition have adequate knowledge in choosing the best tools to solve the given problem. Tools that are purchasable and do not offer free versions or trials are prohibited. There are no restrictions on coding languages that can be used.

Note: Since the use of the Internet and other external resources are permitted in this competition, all information used by competitors must be referenced very carefully. Competitors are not permitted to submit work completed by anyone other than the members of their team. If they decide to recycle their own or someone else's code it must be clearly cited in the presentation. In addition, the competitors also need to clearly explain why and where the recycled code was used in their software. The judges hold the right to ask any team member to describe what a particular section of the code does at any given point during the presentation.

If there is any evidence that competitors are submitting plagiarized work, or that they have significantly edited their code since submittal, the entire team will be eliminated from the competition and their home schools will be notified. Volunteers will monitor each team during the design process to deter teams from cheating and to remind them to cite external resources. However, competitors are expected to act in good faith with the spirit of the competition.

Presentation Rules

Competitors will have a maximum of 20 minutes to present their solutions. All team members must be present and participate in the presentation or be penalized by the judges. Judges then have a maximum of 10 minutes to ask questions. Judges can ask a question at anytime during the presentation. The clock should be stopped during these interruptions.

During the presentation, competitors are expected to run their code on a prepared presentation instance. This instance will be similar but not identical to the development instance, so keep that in mind while creating a solution. A visual representation of what the bot is doing will be displayed during the presentation in order for the judges to assess the capabilities of the code better.

Presentation order shall be determined randomly and shall be announced two hours before the presentations commence. All teams are required to be present at this announcement. Teams are not allowed to switch places in the presentation order, unless there is an emergency. If an emergency arises the competition director must be notified as soon as possible.

A countdown timer will be provided during the presentation that can be viewed by both the judges and the team presenting. The remaining time must be indicated to the competitors 10 minutes, 5 minutes and 1 minute before the end of the allotted time for the presentation to the judges. Time is halted when a judge asks a question during the presentation and when a team member answers a question asked by a judge.

The presentations shall be carried out without an audience. Feedback forms shall be provided to each team following the announcement of winners but prior to the end of CEC.

Judging Matrix

A minimum of three (3) judges (and in any excess, an odd number of judges) are required to assess the problem solving abilities, proposed solution, communication skills and team dynamics of the competitors. Judges in this category should come from a variety of backgrounds including communications, sales and technical or software engineering experience related to the topic. They shall assess presentations based on the following criteria:

Presentation	• Design process • Design justification • Critique of the design • Presentation delivery	20%
Strategy/Algorithm	• Simplicity • Ingenuity • Ability to achieve desired outcome	40%
Code	• Structure • Readability • Code efficiency	30%
Resource Management	• Memory usage efficiency • Program's CPU usage	10%
Total		100%

While the competition lead and judges do have access to your code, is is recommended that you touch on all of these points during your presentation in some capacity in order to convey to the judges a greater understanding of your program's performance in these areas.

Documentation

Error

Method: POST

Request URL: http://cec2019.ca/unloadItem/-1

Header:

token: id1

Response:


```
{
    "type": "ERROR",
    "error": "INVALID_ID",
    "message": "Item ID -1 is not valid. Please select a different item."
}
```


Failure

Method: POST

Request URL: http://cec2019.ca/turn/S

Header:

token: id1

Response:


```
{
    "type": "FAILURE",
    "failure": "TURN_NOT_REQUIRED",
    "message": "You are already facing direction S."
}
```


New Instance

Method: POST

Request URL: http://cec2019.ca/instance

Header:

token: id1

Response:


```
{
    "type": "SUCCESS",
```

```
"payload": {
    "id": "id1",
    "location": {
        "x": 9,
        "y": 9
    },
    "direction": "N",
    "finished": false,
    "timeSpent": 0,
    "itemsLocated": [],
    "itemsHeld": [],
    "itemsBin": [],
    "itemsCollected": [],
    "constants": {
        "ROOM_DIMENSIONS": {
            "X_MIN": 0,
            "X_MAX": 19,
            "Y_MIN": 0,
            "Y_MAX": 19
        },
        "BIN_LOCATION": {
            "ORGANIC": {
                "X": 19,
                "Y": 6
            },
            "RECYCLE": {
                "X": 19,
                "Y": 7
            },
            "GARBAGE": {
                "X": 19,
                "Y": 8
            }
        },
        "TIME": {
            "TURN": 1,
            "MOVE": 1,
            "SCAN_AREA": 4,
            "COLLECT_ITEM": 2,
            "UNLOAD_ITEM": 2
        },
        "TOTAL_COUNT": {
            "ORGANIC": 70,
```

```
                "RECYCLE": 60,
                "GARBAGE": 60
            },
            "BIN_CAPACITY": {
                "ORGANIC": 25,
                "RECYCLE": 15,
                "GARBAGE": 20
            },
            "BIN_COLLECTION_CYCLE": 150,
            "SCAN_RADIUS": 3
        }
    }
}
```


In-Progress Instance

Method: GET

Request URL: http://cec2019.ca/instance

Header:

token: id1

Response:


```
{
    "type": "SUCCESS",
    "payload": {
        "id": "id1",
        "location": {
            "x": 19,
            "y": 6
        },
        "direction": "S",
        "finished": false,
        "timeSpent": 153,
        "itemsLocated": [
            {
                "id": 15,
                "x": 9,
                "y": 8,
                "type": "GARBAGE"
            }
        ],
        "itemsHeld": [
```

```
        {
            "id": 23,
            "x": 9,
            "y": 9,
            "type": "ORGANIC"
        }
    ],
    "itemsBin": [],
    "itemsCollected": [
        {
            "id": 21,
            "x": 9,
            "y": 9,
            "type": "RECYCLE",
            "coveredBy": [
                22
            ],
        },
        {
            "id": 22,
            "x": 9,
            "y": 9,
            "type": "ORGANIC"
        }
    ],
    "constants": {
        "ROOM_DIMENSIONS": {
            "X_MIN": 0,
            "X_MAX": 19,
            "Y_MIN": 0,
            "Y_MAX": 19
        },
        "BIN_LOCATION": {
            "ORGANIC": {
                "X": 19,
                "Y": 6
            },
            "RECYCLE": {
                "X": 19,
                "Y": 7
            },
            "GARBAGE": {
                "X": 19,
```

```
                "Y": 8
            }
        },
        "TIME": {
            "TURN": 1,
            "MOVE": 1,
            "SCAN_AREA": 4,
            "COLLECT_ITEM": 2,
            "UNLOAD_ITEM": 2
        },
        "TOTAL_COUNT": {
            "ORGANIC": 70,
            "RECYCLE": 60,
            "GARBAGE": 60
        },
        "BIN_CAPACITY": {
            "ORGANIC": 25,
            "RECYCLE": 15,
            "GARBAGE": 20
        },
        "BIN_COLLECTION_CYCLE": 150,
        "SCAN_RADIUS": 3
    }
  }
}
```