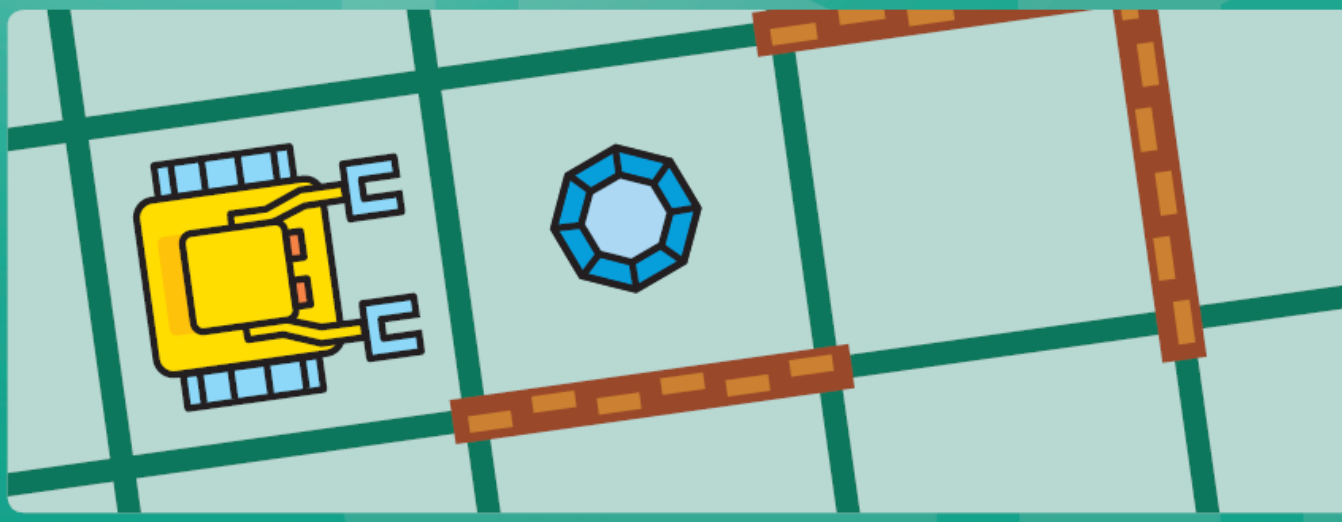
nclab
Public Computing

Learn how to **Think** with

Karel the Robot



Revision October 17, 2013

Why Learn Computer Programming?

Computer programming is lots of fun! Telling a machine what to do and watch it actually perform the task is amazing. Interacting with the computer will teach you *how to think*, which is something that will make your life better in many ways. Programming is all about breaking problems into simpler ones, and thinking how to solve them, which is an extremely useful real life skill.

Why Use a Training Programming Language?

Why spend time with a training programming language if you can learn with the "real stuff" such as C++ or Java? Indeed, these languages are much more powerful than Karel the Robot. But they are not graphical, have complicated syntax, and you will end up doing boring math problems in no time. In contrast to that, Karel the Robot was designed to be graphical, have commands that are easy to type, and it does not contain math.

Why a New Textbook?

Karel the Robot was created at the Stanford University by R.E. Pattis who also wrote the original textbook *Karel the Robot: A Gentle Introduction to the Art of Programming* in the 1980s. His Karel was aimed at university-level students and the syntax was influenced by Pascal, a major programming language of that era. We revived the language by using its original ideas, but our implementation is more oriented towards K-12 students and the syntax is close to Python, a major programming language of today.

About the Author

Dr. Pavel Solin is Professor of Applied and Computational Mathematics at the University of Nevada, Reno. He has been programming computers for 25 years and directing major open source software projects. He is the author of six monographs and many research articles in international journals. Besides this, Dr. Solin enjoys very much working with K-12 teachers and students.

Acknowledgment

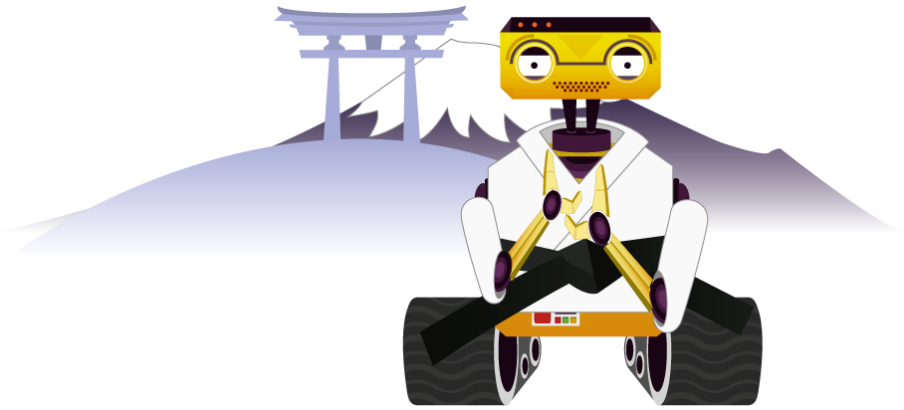
We would like to thank great teachers from NCLab Partner Schools for class-testing Karel, and for providing useful feedback that is helping us to improve the textbook, interactive exercises, and the Karel language itself.

Graphics Design: TR-Design `http://tr-design.cz`

Table of Contents

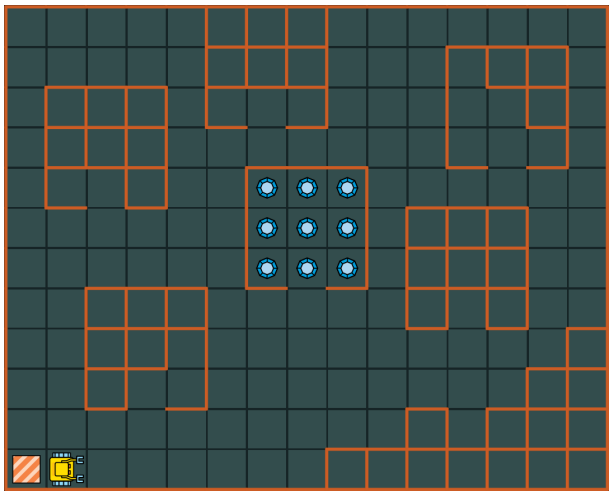
1 Earn Black Belt in Computer Programming!

The best way to learn with Karel is to go through the interactive Karel Course in NCLab, using this textbook as a reference. The course has four differential instruction levels that ensure an exciting and fulfilling learning experience for all students - the White, Yellow, Purple and Black Belts. While the White and Yellow Belts are designed for all students to pass, earning the Black Belt requires a high level of determination.



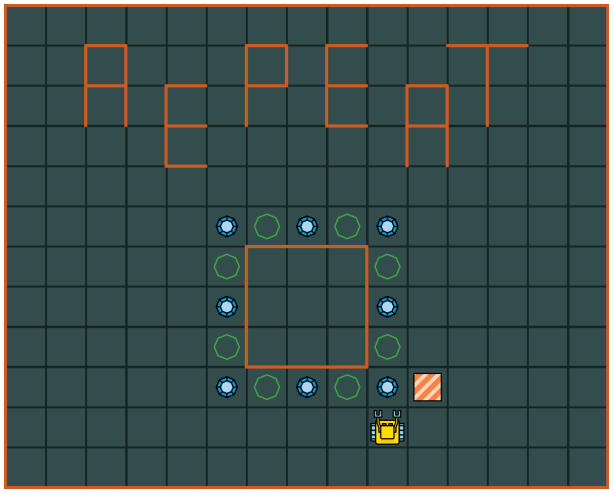
1.1 White Belt

Every student begins his or her journey by completing eight warm up projects in manual mode, earning the White Belt. Sample project *Diamond Mine* from this level is shown below: Karel finds an abandoned diamond mine, and his task is to collect all nine gems and return to his home square in less than 80 seconds:



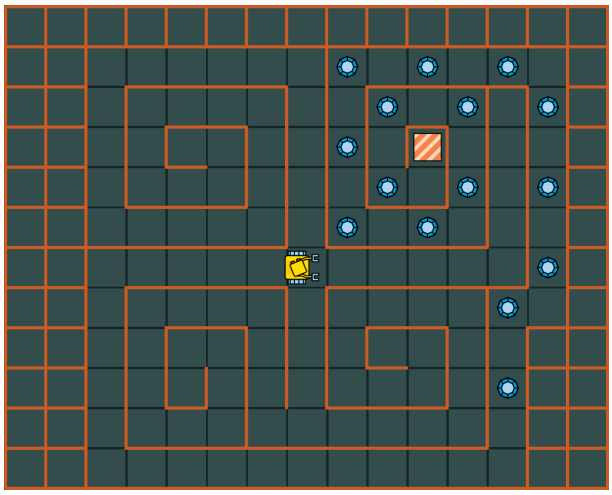
1.2 Yellow Belt

After earning the White Belt, students are invited to pursue the Yellow Belt. This is the core of the course, designed for all students to pass. In order to earn the Yellow Belt, students need to demonstrate knowledge of all major programming concepts, and an ability to write simple computer programs. The following image shows the project *Square Shelf* from this level, where Karel needs to move all gems into trays and return to the home square:



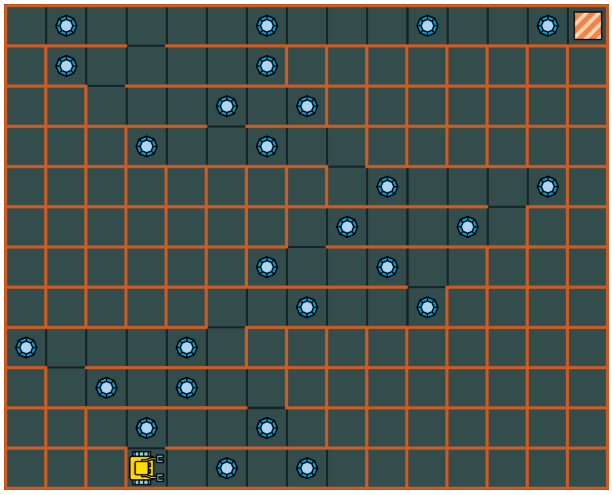
1.3 Purple Belt

In the Purple Belt Level students revisit and reinforce some concepts learned previously, they learn additional advanced concepts, and use them to solve advanced programming projects. The maze shown below corresponds to the project *Curse of the Pharaohs* from this level. Here Karel stands at the cross-section of four tunnels in a pyramid and he faces a random direction - either North, West, South or East. He knows that only the West tunnel leads to the treasure while the remaining three contain deadly traps. He has to be very careful as he does not have a map. The robot needs to reach the end of the tunnel, collecting all gems on the way.



1.4 Black Belt

The Black Belt Level includes challenging programming projects. Their complexity can be illustrated on the sample project *Arcade Game* where Karel needs to collect all gems in an arcade of random proportions.



1.5 Red Belt

Yes, there is a Red Belt Level too! It provides extremely difficult and sometimes open problems for extremely talented young programmers. If you solve one of these problems, we will have a place for you in the NCLab Karel Hall of Fame!

2 Introduction

2.1 Objectives

- Learn basic facts about the Karel language and its history.
- Learn how Karel differs from other programming languages.
- Learn what skills this course will give you.

2.2 Brief history

The educational programming language Karel the Robot was introduced by Richard E. Pattis in his book *Karel The Robot: A Gentle Introduction to the Art of Programming* in 1981. Pattis first used the language in his courses at Stanford University, and nowadays Karel is used at countless schools in the world. The language is named after Karel Čapek, a Czech writer who invented the word "robot" in his 1921 science fiction play R.U.R. (Rossum's Universal Robots). Various implementations of the language that can be downloaded from the web are shown in Fig. 1.

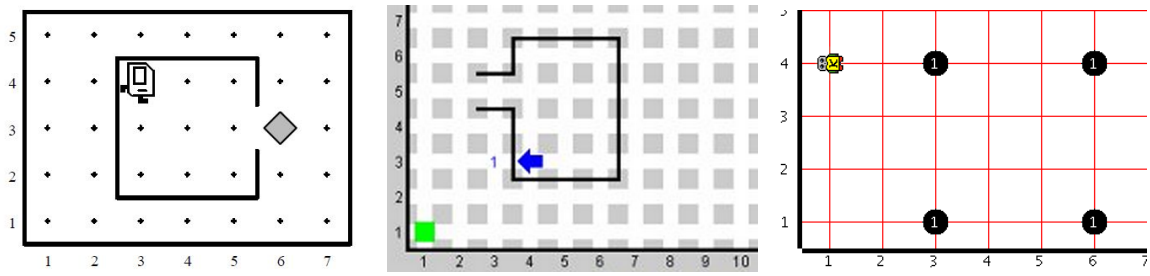


Fig. 1: Various implementations.

The original Karel language was strongly influenced by Pascal, a popular language of the 1980s. Since Pascal is no longer being used today, we refreshed the language and adjusted its syntax to be close to Python, a modern high-level dynamic programming language. Our changes made the language much easier to use. For illustration, compare the original Karel program

```
BEGINNING-OF-PROGRAM

DEFINE turnright AS
BEGIN
    turnleft
    turnleft
    turnleft
END

BEGINNING-OF-EXECUTION
    ITERATE 3 TIMES
    BEGIN
        turnright
        move
    END
    turnoff
END-OF-EXECUTION

END-OF-PROGRAM
```

with its exact NCLab's Karel equivalent

```
def turnright
    repeat 3
        left

repeat 3
    turnright
    go
```

In fact, NCLab's Karel has a built-in command `right` for the right turn, so the above program can be written using just three lines:

```
repeat 3
    right
    go
```

The command `right` was not part of the original Karel language – it was added after a very careful consideration. The main reason for adding it was that it made Karel

more pleasant to watch as he moves through the maze. Without this command, any right turn took three left turns, and as a result, Karel resembled a raging tornado. Of course, orthodox Karel fans can still define and use their own `rightturn` command. We made a few additional changes to the language in order to make it more accessible to kids – beepers were replaced with gems, Karel has a `home` in the maze, and there is a new object `tray` that makes it clear where the robot should put gems. Longer commands were replaced with shorter ones, such as `leftturn` with `left`, `move` with `go`, `pickbeeper` with `get`, and `putbeeper` with `put`. Karel's syntax is virtually identical to Python, with the exception of colons in conditions, loops, and new commands – they were left out since our youngest programmers had difficulty typing them.

2.3 Who is Karel?

Karel is a little robot that lives in a maze and loves to collect gems. He was manufactured with only five simple commands in his memory:

- `go` ... make one step forward.
- `get` ... pick up a gem from the ground.
- `left` ... turn left.
- `right` ... turn right.
- `put` ... put a gem on the ground.

He also has six built-in sensors that allow him to check his immediate surroundings:

- `wall` ... helps the robot detect a wall right ahead of him.
- `gem` ... helps the robot detect a gem beneath him.
- `tray` ... helps the robot detect an empty tray beneath him.
- `north` ... helps the robot detect that he is facing North.
- `home` ... helps the robot detect that he is at home.
- `empty` ... helps the robot detect that his bag with gems is empty.

2.4 What will you learn in this course?

Computer programming skills are highly valued today, and they will be even more valued in the future. Karel is the perfect language for beginners. It will teach you how to design algorithms and write working computer programs without struggling with technical complications of mainstream programming languages. Thanks to its simplicity, you should be done with Karel fairly quickly, and in no time you will be ready to move on to other languages. NCLab offers a Python programming course.

2.5 Is Karel a toy language?

Absolutely not! Despite its playful appearance, Karel features all key concepts of modern procedural programming. Technically speaking, it is a complete Turing machine. As a matter of fact, the complexity of algorithms that you will encounter in this textbook ranges from very simple to very hard.

2.6 How does Karel differ from other programming languages?

The biggest conceptual difference between Karel and mainstream procedural programming languages such as Python, C, C++, Java or Fortran is that *the robot does not know math*. This is because Math is not needed to understand how to design great algorithms and to translate them into efficient computer programs.

3 Launching Karel

3.1 Objectives

- Learn to launch Karel and work with the graphical application.
- Learn that Karel has several modes and how they differ.

The simplest way to launch Karel is to double click on the Programming icon on Desktop and select Karel in the menu. This will launch the application in Programming Mode 1 with a demo program, as shown in Fig. 2.

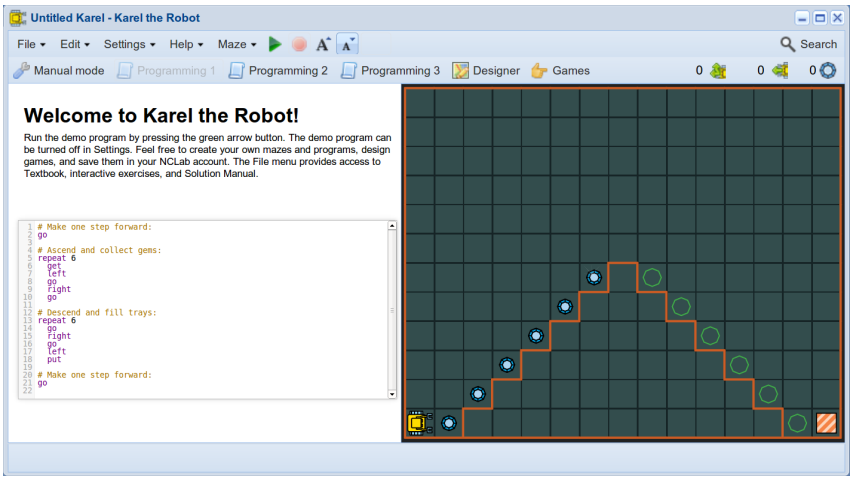


Fig. 2: Launching Karel in Programming Mode 1 with a demo program.

From Programming Mode 1, one can switch to Programming Modes 2 and 3, Manual Mode, Designer, and Game Mode. These modes will be discussed in more detail in Paragraph 3.3.

3.2 Main menu

The application window contains the main menu on top, work area on the left, maze on the right, and status bar on the bottom. The menus are fairly intuitive, so let us explain just a few selected functions. In the *File* menu:

- Under *Learning materials* you will find this textbook, interactive exercises, and instructors have access to solution programs.
- *New* will create a new Karel file.
- *Open* will open an existing Karel file.
- *Save in NCLab* will save your file in your NCLab account.
- *Publish to the web* will create a static HTML link for your project.

The *Maze* menu facilitates operation with mazes, including creating a new random one, duplicating an existing maze, restoring maze to its saved version, and save and remove maze. The *Edit* menu enables operation with code cells and HTML cells (to be discussed in Section 7). In *Settings* one can change Karel's speed, adjust sound preferences, etc.

The green and red buttons are used to run and stop programs, respectively, and the two buttons next to them on the right can be used to increase and decrease font size. The triplet of icons on far right are the operations counter, step counter, and gem counter.

3.3 Karel modes

Karel operates in six modes:

- *Manual Mode.* The robot is controlled using the mouse and five buttons `go`, `get`, `left`, `right`, and `put`. Watch out and do not crash!
- *Programming Mode 1.* Programs are written using six commands `go`, `get`, `left`, `right`, `put`, and `repeat`. The `repeat` command will repeat a given number of times any of the five basic commands, their sequence, or other `repeat` commands.
- *Programming Mode 2.* On top of the commands from Programming Mode 1, programs can contain conditions, conditional loops, and custom commands.
- *Programming Mode 3.* This mode introduces variables, lists, random decisions, basic integer arithmetic, and functions that return values. Karel also gets a GPS device that allows him to determine his position in the maze.
- *Designer* allows the user to create custom mazes.
- *Game Mode* makes it possible to create and play games.

4 Manual Mode

4.1 Objectives

- Review directions on the compass.
- Learn to guide the robot by clicking on buttons.
- Learn about error messages.
- Learn to work with *left* and *right* from the robot's point of view.
- Learn to plan your actions ahead of time.

4.2 Compass

Fig. 3 shows the four directions on the compass: North, South, West, and East.

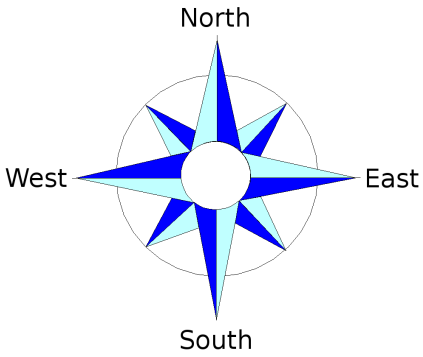


Fig. 3: Four directions on the compass.

4.3 Control buttons

After launching Karel, switch to Manual Mode. The following five buttons will appear in the left panel:

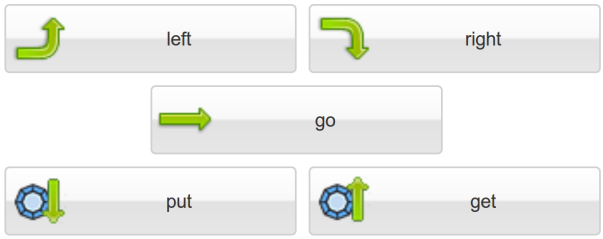


Fig. 4: Karel's buttons in Manual Mode (robot facing East).

Pressing `left` will turn the robot 90 degrees to the left, pressing `right` will turn him 90 degrees to the right. These two buttons never can cause an error, but the others can.

4.4 Error messages

Pressing `go` will advance the robot one step forward. If he crashes into a wall, he will throw an error message:

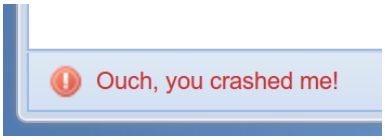


Fig. 5: Error messages appear in the bottom-left corner.

Upon pressing `put`, the robot will reach into his bag and put a gem on the ground. If his bag is empty, he will throw an error message as well. An indicator showing how many gems are in the bag can be found in the upper right corner of the window. Last, pressing `get` makes the robot pick up a gem from the ground. If there is no gem to collect, he will throw an error message.

4.5 Robot's view

When the robot turns, the arrows on the buttons adjust automatically to reflect his view of things. This is illustrated in Fig. 6.

Fig. 6: Karel's buttons in Manual Mode (robot facing West).