

Rover Geometry

Teacher Notes & Answers

7 8 9 10 11 12



TI-Nspire™



Activity



Student



20 min

Teacher: The way any robotic device interacts with the world involves a degree of *error*. Rover is essentially an open loop system which means virtually no feedback is gleaned from its motion.¹ For some situations it can be useful to increase or decrease angles slightly +/- 1 or 2 degrees to obtain better results (allows for slipping).

Problem to Solve

In this activity you will get Rover to draw some simple shapes:

- Square
- Triangle
- Polygon
- 6 pointed Star
- 5 Pointed Star
- House Drawing

Programming

Press the HOME key:  on

Start a new document.

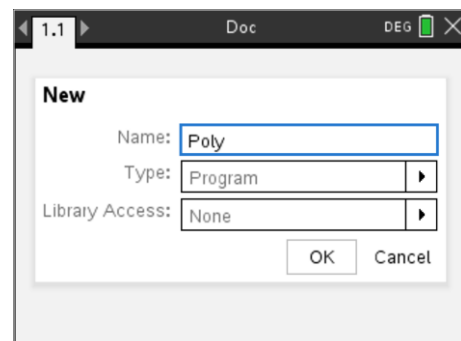
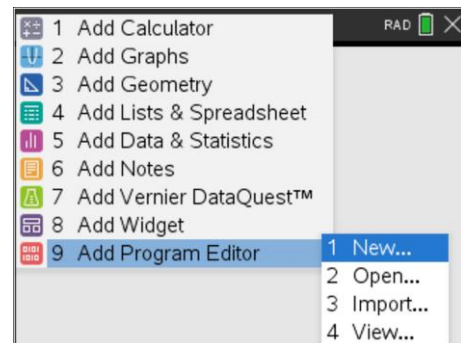
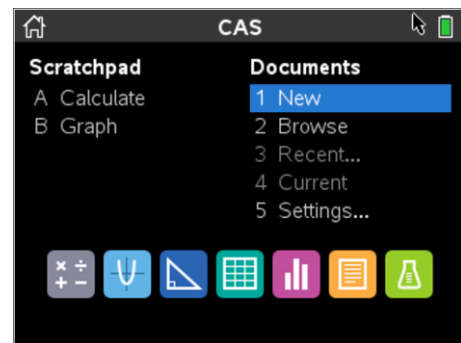
If you are prompted to save your existing document, select NO.

There are many applications to choose from on the TI-Nspire calculator. We will be starting with the Programming application.

Select option 9: **Add Program Editor > New**

When prompted, type the name of the program.

Program Name: Poly

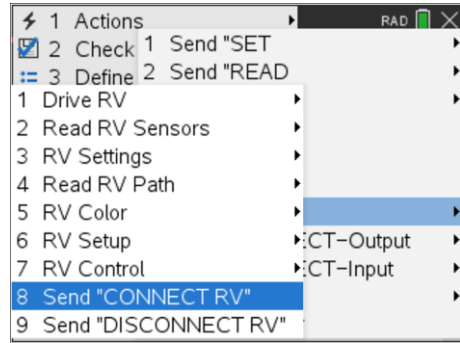


¹ Feedback: Rover actually contains a digital gyroscope, so it does get some feedback.

The first task for the program is to alert Rover that the calculator is attached so that the communication lines are open. This is a built in programming instruction.

Press: **menu** > **Hub** > **Rover** > **Send "CONNECT RV"**

Press: **enter** [This accepts the command and inserts a new line]



The next step is to send a driving command to Rover.

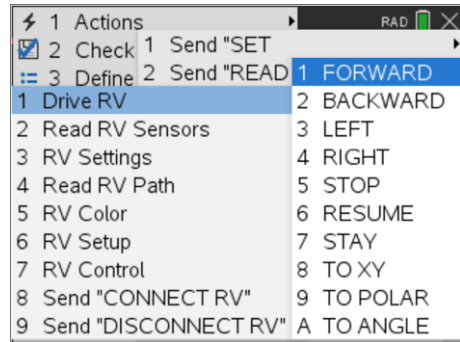
Press: **menu** > **Hub** > **Rover** > **Drive RV** > **Forward**

Before pressing the Enter key, insert a number between 1 and 2.

Your command will look like:

Send "RV FORWARD 1.2"

Press **enter** to create another blank line.

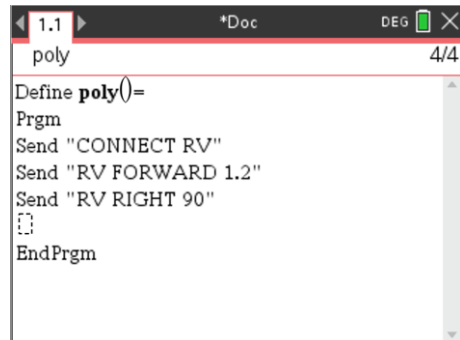


The next command will instruct Rover to turn right or left.

Press: **menu** > **Hub** > **Rover** > **Drive RV** > **RIGHT**

An angle needs to be entered after the "RIGHT" command.

To draw a square use: 90



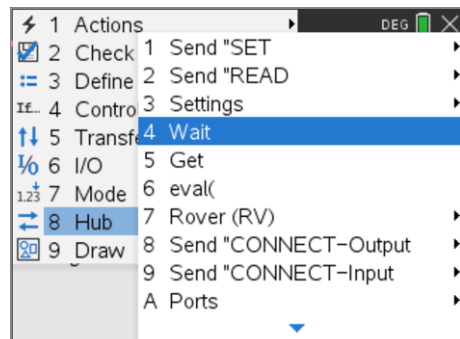
The brain behind Rover is the TI-Innovator. The TI-Innovator can only hold a relatively small number of commands in its buffer. To prevent loss of commands insert an occasional 'wait' command and specify an amount of time (seconds). The calculator will stop sending commands during this time allowing Rover to catch up.

A range of commands are available to detect what Rover is up to, however a simple solution is to measure or estimate how long the current commands will take to physically execute and have the calculator program 'wait' before sending additional commands.

The time taken to drive forwards a short distance and then turn takes approximately 3 seconds.

Press: **menu** > **Hub** > **Wait**

Enter a wait time of 3 seconds: Wait 3



The commands highlighted opposite need to be repeated 4 times to draw a square.

The commands could simply be copied (Copy =Ctrl + C) and pasted (Paste = Ctrl + V) repeatedly. There is however a programming alternative which is much more efficient.

For $i,1,4$
Instructions
Endfor

In this example the 'i' is a counter that starts at 1 and finishes when the instructions have been executed 4 times.

Hold down the SHIFT key to highlight the commands to be repeated.

While the commands are highlighted:

Press:  > **Control** > **For...EndFor**

The syntax in the For command is as follows:


For *variable, start, end*

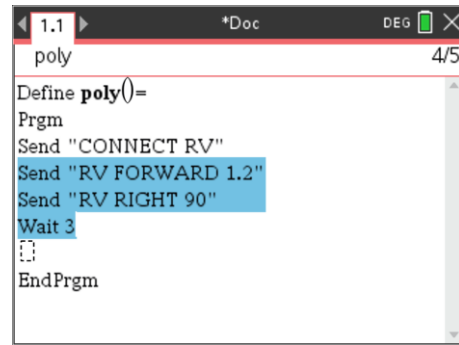
The variable can be given any name, i, j, n are all common.

The first time the commands are executed $i = 1$; the second time the commands are executed $i = 2$... the last time $i = 4$

The program is now ready to be saved and Rover is ready to drive.

Press  then 


Place Rover on the driving mat with a clear space in front for Rover to drive, then press .



```

1.1 | *Doc | DEG | 4/5
poly
Define poly()=
Prgm
Send "CONNECT RV"
Send "RV FORWARD 1.2"
Send "RV RIGHT 90"
Wait 3
EndPrgm

```



```

1.1 | *Doc | DEG | 2/7
* poly
Define poly()=
Prgm
Send "CONNECT RV"
For i,1,4
Send "RV FORWARD 1.2"
Send "RV RIGHT 90"
Wait 3
EndFor
EndPrgm

```



```

1.1 | 1.2 | *Doc | DEG
poly()

```

Experimentation

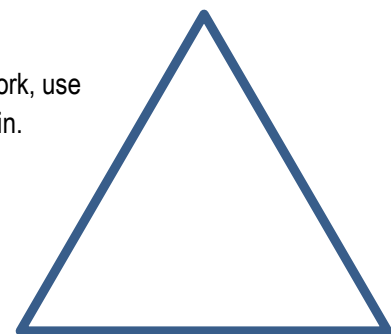
It's now time to experiment. You can edit the program to change how far Rover travels forward. You can also put a piece of paper beneath Rover and a felt-tip pen in the holder to get Rover to draw whilst driving.

Challenge 1

Edit the poly program so that Rover draws a triangle. If your first attempt doesn't work, use the result to think about what changes you may still need to make and then try again.

What type of triangle did Rover draw?

Teacher: A square is an easy polygon to start, however students usually misinterpret the 90° turn. Students need to think about the external angles, so when they program Rover to draw a triangle they often use 60° as the turning angle. As it 'turns' out, the external angles are easy to work with. Students can think about Rover's start and finish position and realise that after making just three turns Rover will have done a complete 360° turn. This thinking leads to the external angles being 120° each. Students should also think about the type of triangle they are drawing (equilateral), all other triangles will take much more working out.

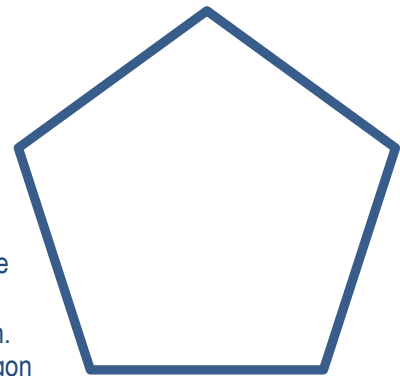


Challenge 2

Edit the poly program so that Rover draws a pentagon.

Think about how the angles are calculated for the program and what Rover is actually doing when it turns. We are looking to generalise the program so that Rover can do any regular polygon.

Teacher: Following on from the triangle, if students have thought about the external angles, all other polygons will be easier. In each case Rover will complete a 360° turn, therefore for an 'n' sided polygon each turn will be equal to $360/n$. From a geometrical perspective this means the internal angles will be $180 - 360/n$. Students should also recognise that once again they are drawing a regular polygon (pentagon) and that irregular polygons would require considerable more calculations.



Challenge 2 - Extension

Rather than edit the program each time, it is much more convenient to simply enter the number of sides required. A Request command provides a text prompt (for the user) and expects a value to be entered.

Navigate to the CONNECT RV command and insert a line immediately below then add in the Request command.

Press: **[menu]** > I / O > Request

Inside the quotation marks (Press **[?]** or Ctrl + x) enter the text prompt. Outside the quotation marks use a comma (,) followed by the variable name. (n)

Change the number of loops in the **For** command to go up to n.

In the turn right command change the angle to `eval(expression)` where expression represents a mathematical formula, involving 'n' that determines the size of the angle that Rover needs to turn.

Once you are happy with your program, run it and see if it works! You know what to do if it doesn't.

```

1.1 1.2 *Doc DEG X
* poly 5/8
Define poly()=
Prgm
Send "CONNECT RV"
Request "Sides",n
For i,1,n
Send "RV FORWARD 1.2"
Send "RV RIGHT eval( )"
Wait 3
EndFor

```

Challenge 3

You can copy your program or start a new one. Your challenge this time is to write a program that draws a 6 pointed star. You should start by drawing a diagram; think about the angles and the collection of commands that need to be repeated to achieve the outcome.



The quick way to duplicate a program is to use the **[menu]** key and select **Actions** then **Create copy...**

You can simply type in the name for the new program 'star6' and start editing. The new program will appear alongside your existing program. Press Ctrl + 6 to split these two applications onto two pages.

Teacher: A drawing of the star has not been provided here so as to encourage different approaches to the drawing. Students may typically draw two overlapping triangles leading them to the idea that they will somehow use the triangle program from the first challenge. It is actually easier to think only about the outline. Students are encouraged to draw diagrams and calculate angles based on their knowledge of geometry.

Challenge 4

Your challenge this time is to write a program that draws a 5 pointed star. You should start by drawing a diagram. This one might take a bit more thinking!

Can you extend your program to create an 'n' pointed star?

Teacher: Interesting that the 5 point star is listed after the 6 point. This is because the 6 point star (outline) is easy to produce by overlapping two equilateral triangles. This makes calculating angles much easier. The 5 point star however is a little more complicated.

Just like the 6 point star, students may choose to drive around the outline of the star. There are numerous ways that students may calculate the angles for the 5 point star.

Students can start with some basics:

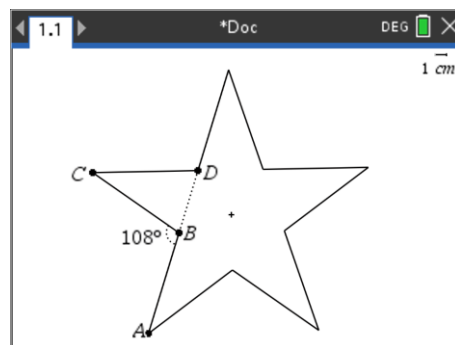
$$\angle ABC + \angle DBC = 180^\circ$$

$$2 \times \angle DBC + \angle BCD = 180^\circ$$

What about:

$$5 \times \angle ABC - 5 \times \angle BCD = 360^\circ$$

The above set of relationships is sufficient to determine the necessary angles. Notice that the third condition really comes from thinking about Rover's orientation at the start and end of the journey.



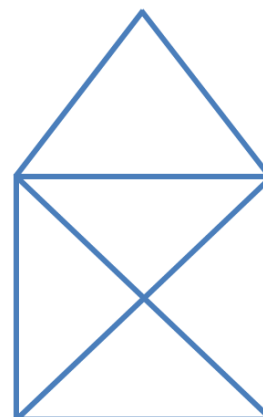
Challenge 5

A popular challenge is to draw the shape shown opposite (house) so that each line is drawn just once and you are not allowed to take your pen of the page.

Try drawing it by hand first, then write the code so that Rover can also draw it.

What challenges do you face in trying to draw this object?

Teacher: This of course is the common drawing problem provided to students. It should be noted that the drawing will always start at the bottom and end at the bottom (opposite vertex).



Challenge 6

The Konigsberg bridge problem is very famous. The challenge is to visit each bank and each island and cross every bridge just once.

Think of Rover trying to navigate around the North and South bank, bridges and islands. Can you write a program that crosses each bridge just once?

Remember – Rover can't swim!

Teacher: A great opportunity to briefly introduce Graph Theory!

