Kapitel 6: Koordinater — Tillämpning: Slumpvandring

Översikt: Upptäckter med slumptal kan leda till fascinerande observationer. Denna tillämpning ger dig möjlighet att utforska mer ovanliga saker om sannolikhet och att programmera Rover att flytta runt på ett rutnät.

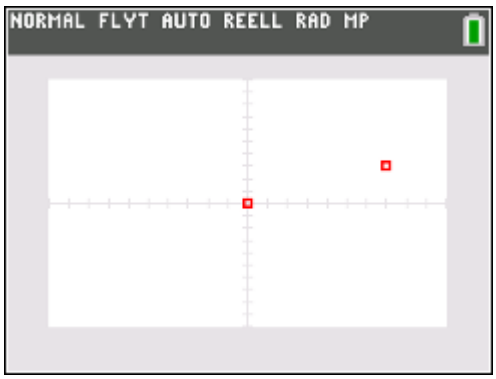
Slumpvandring är ett experiment med datorprogrammering. Denna tillämpning knyter samman flera olika programmeringsfärdigheter.

Syfte:

- Använda koordinatrörelse för att simulera en slumpvandring
- Använda räkneverk och ackumulatorer i ett program
- Använda sammansatta villkor med **inte** och **och**.

Problemet:

Antag att din stads olika gator är upplagda i ett fyrkantigt rutmönster och att din skola ligger vid (0, 0) i rutnätet. Ditt hem är på (7, 3), som representerar 7 kvarter öster och 3 kvarter norr om skolan. (Överväg gärna förändringar hos denna position.) Bilden till höger är en statistisk plottning av dessa två punkter.



Från skolan går du ett kvarter i slumpmässig riktning (norr, söder, öster eller väster). Vid varje korsning går du sedan ett kvarter i slumpmässig riktning igen. Kommer du någonsin att komma hem? Hur många kvarter kommer du vandra innan du kommer hem? Kommer du sluta på grund av utmattning innan du kommer hem?

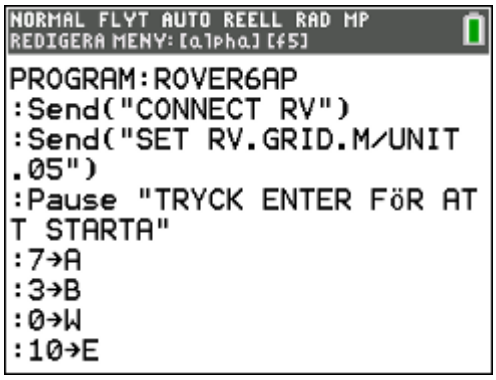
Planering är en viktig del av kodningen. Tänk på vad Rover kan göra och vad ditt programmeringsspråk kan göra.

Tänk på att när man arbetar med slumptal så är vi i händerna på maskinen. Det kan ta lång tid för Rover att nå hem, så vi ska lägga till en bestämmelse om att Rover endast får förflytta sig ett begränsat antal kvarter innan du slutar.

Lärarkommentar: Förbered eleverna så att programmet kan använda en **Input-sats** för koordinaterna (7, 3) för hemmet. Använd variabler för den positionen från början. I själva verket är det bättre att använda variabler så mycket som möjligt (snarare än specifika värden). Användningen av variabler gör det möjligt att enkelt ändra värden eller modifiera programmet att använda **Input**-satser snarare än **Lagra**-satser för att initiera simuleringens tillstånd.

1. Starta programmet med CONNECT RV-kommandot. Ställ in Rovers rutnätsstorlek till 5 cm med
   Send ("SET RV. GRID. M/UNIT. 05 ").

   Tänk på att det här kommandot finns i **prgm > Hub > Rover (RV)... > RV setup...** och det förändrar Rovers "enhet" för rörelse (används i **FORWARD 1**) från 10 cm till 5 cm, vilket möjliggör fler rutnätspunkter på ett mindre utrymme.

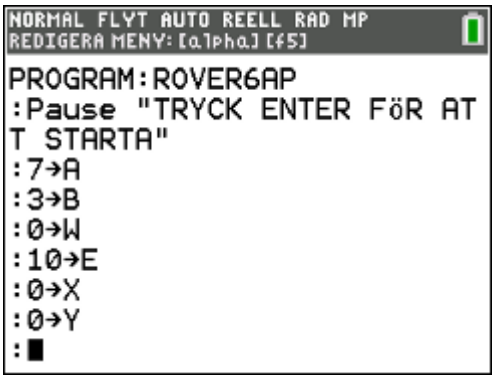


Initiera variabler

2. Spara hemkoordinaterna (7, 3) och startnumret för passerade kvarter (0) i variablerna A, B respektive W. Variabeln W kommer att användas för att hålla reda på antalet kvarter som Rover ”går” och kommer att användas för att stoppa programmet om Rover går för långt. Rover slutar av utmattning när den har gått ett angivet antal kvarter.

   Låt oss använda 10 kvarter som ett värde för utmattningssträckan. Använd variabeln E för detta värde: 10→E.

3. Rover kommer hem när dess position, som vi kallar (***X***, ***Y***), är (7, 3). Initiera båda variablerna, ***X*** och ***Y***, att vara noll.



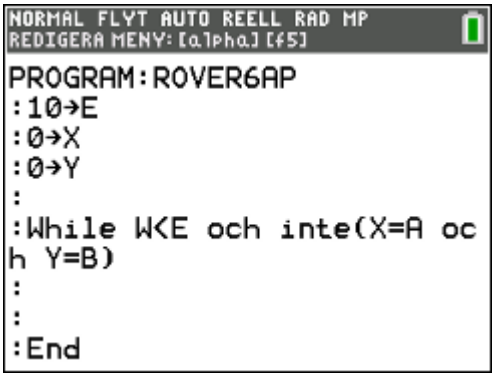
Huvudloopen

4. Huvudloopen hos programmet består av en **While**-loop med två villkor som ska adresseras. Villkoren är ”när Rover är hemma” (när X = A och Y = B) eller slutar av utmattning (när W = E). Rover slutar när variabeln W (antalet kvarter som passerats) är lika med det värde som lagras i E.

   While-loopen fortsätter så länge som villkoren är falska så vi ställer upp motsatta villkor.
   While-loopen blir då: **While W<E och inte(X=A och Y=B)**

   Kom ihåg att fasta värden som är relaterade till problemet lagras i **E**, **A**, och **B**.

   inte( ) används för att försäkra att programmet fortsätter så länge som Rover inte är ”**hemma**”.



5. Kom ihåg att inkludera en **End**-sats för loopen.

Inne i loopen

6. Tillväxten **W** (antalet passerade kvarter): **W+1→W**.

7. Välj riktning slumpmässigt (norr, söder, öst, eller väst):
   - Rover’s **TO ANGLE**-kommando vrider Rover till en “absolute” riktning: 0 är öst, 90 är norr, 180 är väst, och 270 är syd
   - **slumpHel(0,3)** ger ett slumpmässigt heltal: 0, 1, 2, eller 3.
   - Multiplicera detta värde med 90 för att få 0, 90, 180, eller 270.
   - Satsen för att få en slumpmässig riktning **D** är:

   90*slumpHel(0,3)→D

8. Vrid Rover till vinkeln **D**: **Send("RV TO ANGLE eval(D)")**.
9. Förflytta Rover framåt 1 enhet (1 kvarter i vår simulering): **Send("RV FORWARD 1")**.
10. Uppdatera Rover's position i programmet:
    Om Rover går åt norr, öka då **Y** med 1
    Om Rover går åt öster, öka då **X** med 1
    Om Rover går åt söder, minska då **Y** med 1
    Om Rover går åt väster, minska då **X** med 1

Inkludera några **Wait-**kommandon för att hålla programmet synkroniserat med Rover´s rörelser. Vridning tar tid och att gå framåt tar tid. **Wait-**tiderna beror på vridningsvinkeln och tillryggalagd sträcka så att du kan behöva experimentera med **Wait-v**ärdena.

Efter loopen

Loopen avslutas på ett av två sätt:

- Om **W**=**E**, så slutar Rover sin vandring på grund av utmattning. Visa då "ROVER SLUTAR" på räknarskärmen.
- Om **X**=**A** och **Y**=**B**, är Rover hemma. Visa då meddelandet "ROVER KOM NU HEM" på räknarskärmen.

Lärarkommentar:
Exempel på lösning: Eleverna kan testa sin kod utan en Rover. **Send**-kommandon kommer inte att göra någon skada. Den första DISP-satsen i koden nedan anger var Rover är i sitt koordinatsystem även om Rover inte är tillgänglig. Väntetiderna (**Wait**) kan behöva justeras och kan tas bort för att köra programmet snabbare utan Rover.

```
ClrHome
Disp "ROVER KAP6 APP"
Send("CONNECT RV")
Send("SET RV.GRID.M/UNIT .05")
Pause "TRYCK ENTER FÖR ATT STARTA"
7→A
3→B
0→W
10→E
0→X
0→Y
While W<E och inte(X=A och Y=B)
W+1→W
90slumpHel(0,3)→D
Send("RV TO ANGLE eval(D)")
"Wait 1
Send("RV FORWARD 1")
"Wait .5
```

```
If D=0:X+1→X
If D=90:Y+1→Y
If D=180:X-1→X
If D=270:Y-1→Y
Disp {X,Y}
End
If W=E:Disp "ROVER SLUTAR"
If X=A och Y=B:Disp "ROVER KOM NU HEM"
```

Möjliga tillägg:

- Tänd lysdioden COLOR LED i olika färger för olika riktningar (endast under rörelse och inte under vridning). Timingen är här avgörande.
- Lägg till ljud för specialeffekter.
- Lägg till ett test för att se om Rover någonsin är tillbaka i skolan efter att den börjat röra sig. Detta kan vara ett annat villkor för att avsluta programmet och som kan läggas till While-villkoret (men inte i början eftersom loopen aldrig skulle komma igång).
- Lägg till kod som håller reda på och visar det totala antalet kvarter som Rover har rört sig och slutligen avståndet (fågelvägen) från utgångspunkten.
- Överväg begränsningar så att Rover bara kan röra sig österut eller norrut. Hur påverkar detta logiken i programmet? (*Tips*: titta på While-villkoret.)
- Bonus 1: hur långt från skolan (fågelvägen) är Rover i slutet av programmet?
- Bonus 2: Vad var det maximala avståndet från skolan? Hemifrån?