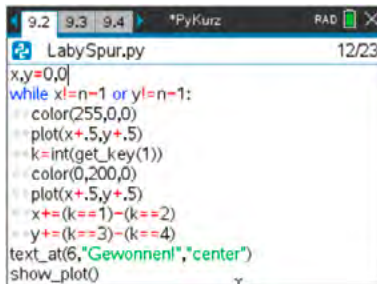


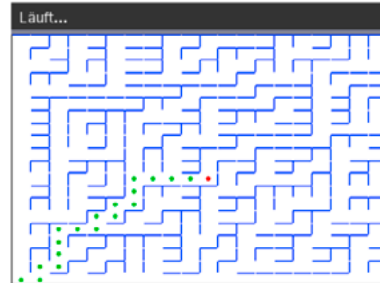
Der Weg durchs Labyrinth

LabSpur.py, LabSpur1.py

Jetzt wollen wir noch einen Punkt in den Irrgarten stellen, den wir durch den Irrgarten führen. Dazu müssen wir zuerst ein zusätzliches Modul `from ti_system import *` an die Spitze des Programms laden und den letzten Befehl `show_plot` vorerst löschen.



```
x,y=0,0
while x!=n-1 or y!=n-1:
  color(255,0,0)
  plot(x+.5,y+.5)
  k=int(get_key(1))
  color(0,200,0)
  plot(x+.5,y+.5)
  x+=(k==1)-(k==2)
  y+=(k==3)-(k==4)
text_at(6,"Gewonnen!","center")
show_plot()
```

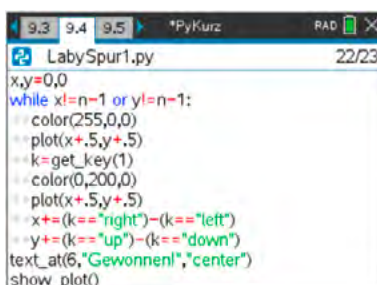


- Der Spieler startet im Ursprung ganz in Rot (links unten) und durchläuft dann eine while-Schleife bis die obere rechte Ecke $(n - 1, n - 1)$ erreicht wird.
- Die jeweils aktuelle Position des Spielers wird durch den roten Punkt angezeigt. Dann wird ein Tastendruck für die Weiterbewegung erwartet.
- Sobald dieser erfolgt, wechselt der Spieler seine Farbe in Grün und bewegt sich in Abhängigkeit von den gedrückten Steuerungstasten: Mit den Tasten 1, 2, 3 und 4 wird der Punkt in die Richtungen \rightarrow , \leftarrow , \uparrow und \downarrow geführt.
Die Programmzeile `x+=(k==1)-(k==2)` erhöht oder erniedrigt den x -Wert, je nachdem welche Taste gedrückt wurde. Analoges geschieht mit dem y -Wert. Anschließend wird die Schleife wieder durchlaufen.
- Die Schleife wird beendet, wenn der Spieler sein Ziel erreicht hat und der Erfolg wird angezeigt.

Irrgärten bilden eine binäre Baumstruktur und können einfach durchlaufen werden. Wie kannst du das tun? Erkläre auch die Bezeichnung *binärer Baum*.

In dieser Version kann der Spieler auch „durch die Wand“ gehen, es gibt keinen Programmteil, der dies verbietet. Man kann einen „Wandcheck“ einbauen, das ist aber ein Stück mühsamer und verlangsamt den Programmablauf. Du kannst aber das Spiel herausfordernder machen, indem du `-(k==2)` und `-(k==4)` aus dem Code nimmst. Damit kann der Spieler keinen Schritt zurück machen.

Die Führung des Punktes wird intuitiver, wenn du anstelle der Zifferntasten die Pfeiltasten verwendest. Kleine Änderungen im Programmcode machen das leicht möglich (LabSpur1.py).



```
x,y=0,0
while x!=n-1 or y!=n-1:
  color(255,0,0)
  plot(x+.5,y+.5)
  k=get_key(1)
  color(0,200,0)
  plot(x+.5,y+.5)
  x+=(k=="right")-(k=="left")
  y+=(k=="up")-(k=="down")
text_at(6,"Gewonnen!","center")
show_plot()
```

