

# Masterclass programmeren op de GR TI-84 (les 5)

Kevin van As

August 6, 2016

# Recap!

We hebben gekeken naar:

- Advanced I/O
  - Menus
  - Output
- Precisie en afrondingsfouten
- Debugging
- Eigen functions definiëren

# Recap!

We hebben gekeken naar:

- Advanced I/O
  - Menus
  - Output
- Precisie en afrondingsfouten
- Debugging
- Eigen functions definiëren

# Recap!

We hebben gekeken naar:

- Advanced I/O
  - **Menus**
  - `Output`
- Precisie en afrondingsfouten
- Debugging
- Eigen functions definiëren

# Recap!

We hebben gekeken naar:

- Advanced I/O
  - Menus
  - Output
- Precisie en afrondingsfouten
- Debugging
- Eigen functions definiëren

# Recap!

We hebben gekeken naar:

- Advanced I/O
  - Menus
  - Output
- Precisie en afrondingsfouten
- Debugging
- Eigen functions definiëren

# Recap!

We hebben gekeken naar:

- Advanced I/O
  - Menus
  - Output
- Precisie en afrondingsfouten
- Debugging
- Eigen functions definiëren

# Recap!

We hebben gekeken naar:

- Advanced I/O
  - Menus
  - Output
- Precisie en afrondingsfouten
- Debugging
- Eigen functions definiëren



# Recap!

We hebben gekeken naar:

- Advanced I/O
  - Menus
  - Output
- Precisie en afrondingsfouten
- Debugging
- Eigen functions definiëren

# Vandaag...

GAMES!

# Vandaag...

GAMES!

Wat hebben we nodig?

# Vandaag...

## GAMES!

Wat hebben we nodig?

- Graphics
- Real-time control
- Multiplayer (?)
- Een goed origineel idee!

# Outline

- 1 Graphics
  - Friendly window
  - Basic drawing functions
  - Sprites
- 2 Dynamic input
  - getKey
  - wait for input
  - Memory Leaks
- 3 Games!
  - Game Loop
  - Towards multiplayer...
- 4 Exercises
  - Exercises
  - Further Reading

# Outline

- 1 Graphics
  - Friendly window
  - Basic drawing functions
  - Sprites
- 2 Dynamic input
  - getKey
  - wait for input
  - Memory Leaks
- 3 Games!
  - Game Loop
  - Towards multiplayer...
- 4 Exercises
  - Exercises
  - Further Reading

# Friendly window

- Om iets te tekenen, hebben we coördinaten nodig.
- Een mooi coördinatenstelsel is handig!
- Kies zodat elke pixel “1” waard is.
  - Coördinaten  $x \in [0, 94]$  en  $y \in [0, 62]$
- See also:  
<http://tibasicdev.wikidot.com/friendly-window>

# Friendly window

- Om iets te tekenen, hebben we coördinaten nodig.
- Een mooi coördinatenstelsel is handig!
- Kies zodat elke pixel “1” waard is.
  - Coördinaten  $x \in [0, 94]$  en  $y \in [0, 62]$
- See also:  
<http://tibasicdev.wikidot.com/friendly-window>



# Friendly window

- Om iets te tekenen, hebben we coördinaten nodig.
- Een mooi coördinatenstelsel is handig!
- Kies zodat elke pixel “1” waard is.
  - Coördinaten  $x \in [0, 94]$  en  $y \in [0, 62]$
- See also:  
<http://tibasicdev.wikidot.com/friendly-window>

# Friendly window

- Om iets te tekenen, hebben we coördinaten nodig.
- Een mooi coördinatenstelsel is handig!
- Kies zodat elke pixel “1” waard is.
  - Coördinaten  $x \in [0, 94]$  en  $y \in [0, 62]$
- See also:

<http://tibasicdev.wikidot.com/friendly-window>

```
:ZStandard  
:84→Xmin  
:72→Ymax  
:ZInteger
```

# Friendly window

- Om iets te tekenen, hebben we coördinaten nodig.
- Een mooi coördinatenstelsel is handig!
- Kies zodat elke pixel “1” waard is.
  - Coördinaten  $x \in [0, 94]$  en  $y \in [0, 62]$
- See also:

<http://tibasicdev.wikidot.com/friendly-window>



```

ZOOM MEMORY
3↑Zoom Out
4:ZDecimal
5:ZSquare
6:ZStandard
7:ZTrig
8:ZInteger
9↓ZoomStat
    
```

```

:ZStandard
:84→Xmin
:72→Ymax
:ZInteger
    
```

# Friendly window

- Om iets te tekenen, hebben we coördinaten nodig.
- Een mooi coördinatenstelsel is handig!
- Kies zodat elke pixel “1” waard is.
  - Coördinaten  $x \in [0, 94]$  en  $y \in [0, 62]$
- See also:

<http://tibasicdev.wikidot.com/friendly-window>



```

X/Y T/θ U/V/W
1: Xmin
2: Xmax
3: Xscl
4: Ymin
5: Ymax
6: Yscl
7: Xres
    
```

```

VARS Y-VARS
1: Window...
2: Zoom...
3: GDB...
4: Picture...
5: Statistics...
6: Table...
7: String...
    
```

```

:ZStandard
:84→Xmin
:72→Ymax
:ZInteger
    
```

# Friendly window

- Om iets te tekenen, hebben we coördinaten nodig.
- Een mooi coördinatenstelsel is handig!
- Kies zodat elke pixel “1” waard is.
  - Coördinaten  $x \in [0, 94]$  en  $y \in [0, 62]$
- See also:  
<http://tibasicdev.wikidot.com/friendly-window>



$\frac{X}{Y}$  T/ $\theta$  U/V/W  
 1: Xmin  
 2: Xmax  
 3: Xsc1  
 4: Ymin  
 5: Ymax  
 6: Ysc1  
 7: Xres

```

WARS Y-VARS
Window...
2: Zoom...
3: GDB...
4: Picture...
5: Statistics...
6: Table...
7: String...

```

```
:ZStandard
:84→Xmin
:72→Ymax
:ZInteger
```

## PRGM ØFWINDOW

- Het is handig om een subprogramma te maken om een friendly window te activeren.
- Maak dit PRGM na!



```
PROGRAM: ØFWINDOW
:FnOff
:GridOff
:AxesOff
:ZStandard
:84→Xmin
:72→Ymax
:ZInteger
:ClrDraw
```

# Outline

- 1 Graphics
  - Friendly window
  - Basic drawing functions
  - Sprites
- 2 Dynamic input
  - getKey
  - wait for input
  - Memory Leaks
- 3 Games!
  - Game Loop
  - Towards multiplayer...
- 4 Exercises
  - Exercises
  - Further Reading

# Schermgrootte & Draw line

- TI-84: 96x64 pixels
  - Om te tekenen:  $x \in [0, 94]$  en  $y \in [0, 62]$
- In het [DRAW]-menu staan functies om te tekenen.
- `PRGMOUTLINE` kleurt alle buitenste pixels.

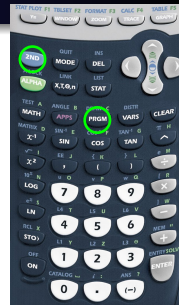


# Schermgrootte & Draw line

- TI-84: 96x64 pixels
  - Om te tekenen:  $x \in [0, 94]$  en  $y \in [0, 62]$
- In het [DRAW]-menu staan functies om te tekenen.
- `PRGMOUTLINE` kleurt alle buitenste pixels.

# Schermgrootte & Draw line

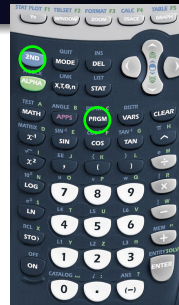
- TI-84: 96x64 pixels
  - Om te tekenen:  $x \in [0, 94]$  en  $y \in [0, 62]$
- In het [DRAW]-menu staan functies om te tekenen.
- `PRGMOUTLINE` kleurt alle buitenste pixels.



```
DRAW POINTS STO
1:ClrDraw
2:Line(
3:Horizontal
4:Vertical
5:Tangent(
6:DrawF
7:Shade(
```

# Schermgrootte & Draw line

- TI-84: 96x64 pixels
  - Om te tekenen:  $x \in [0, 94]$  en  $y \in [0, 62]$
- In het [DRAW]-menu staan functies om te tekenen.
- **PRGMOUTLINE** kleurt alle buitenste pixels.

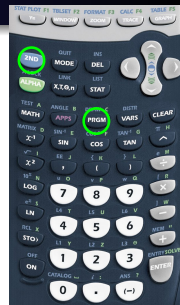


```
PROGRAM:OUTLINE
:ZStandard
:84→Xmin
:72→Ymax
:ZInteger
:Line(0,0,0,62)
:Line(0,0,94,0)
:Line(0,62,94,62)
:Line(94,0,94,62)
```

```
DRAW POINTS STO
1:ClrDraw
2:Line(
3:Horizontal
4:Vertical
5:Tangent(
6:DrawF
7:Shade(
```

# Let's draw!

- Alles om te tekenen staat in het [DRAW]-menu.
- Teken eens:
  - 1 Een lijn → een driehoek
  - 2 Een cirkel
  - 3 Een smily
  - 4 Wat je wilt!
- Let op hoe snel/handig alle functies zijn.
- Dit hoeft niet in een `PRGM`: kan gewoon op het hoofdscherm.



```
:ZStandard
:84→Xmin
:72→Ymax
:ZInteger
```

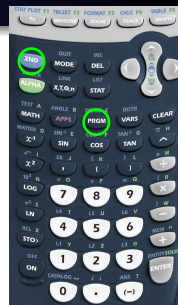
```

DRAW POINTS STO
1:ClrDraw
2:Line(
3:Horizontal
4:Vertical
5:Tangent(
6:DrawF
7:Shade(

```

# Let's draw!

- Alles om te tekenen staat in het [DRAW]-menu.
- Teken eens:
  - 1 Een lijn → een driehoek
  - 2 Een cirkel
  - 3 Een smily
  - 4 Wat je wilt!
- Let op hoe snel/handig alle functies zijn.
- Dit hoeft niet in een PRGM: kan gewoon op het hoofdscherm.



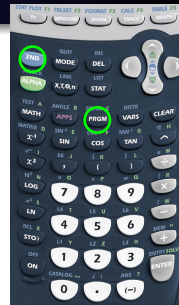
```
:ZStandard
:84→Xmin
:72→Ymax
:ZInteger
```

```
:Line(5,5,35,35)
:Circle(20,20,10)
:DrawF X^1.1
:Shade(X,X²,0,10,1,1)
:Shade(X,X²,10,20,1,2)
:Shade(X,X²,20,30,1,4)
:Shade(X,X²,30,40,2,4)
:Shade(X,X²,40,99,3,7)
```

```
DRAW POINTS STO
1:ClrDraw
2:Line(
3:Horizontal
4:Vertical
5:Tangent(
6:DrawF
7↓Shade(
```

# Let's draw!

- Alles om te tekenen staat in het [DRAW]-menu.
- Teken eens:
  - 1 Een lijn → een driehoek
  - 2 Een cirkel
  - 3 Een smily
  - 4 Wat je wilt!
- Let op hoe snel/handig alle functies zijn.
- Dit hoeft niet in een PRGM: kan gewoon op het hoofdscherm.



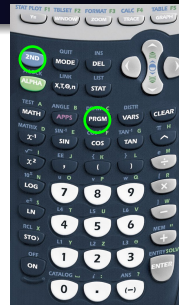
```
:ZStandard
:84→Xmin
:72→Ymax
:ZInteger
```

```
:Line(5,5,35,35)
:Circle(20,20,10)
:DrawF X^1.1
:Shade(X,X²,0,10,1,1)
:Shade(X,X²,10,20,1,2)
:Shade(X,X²,20,30,1,4)
:Shade(X,X²,30,40,2,4)
:Shade(X,X²,40,99,3,7)
```

```
DRAW POINTS STO
1:ClrDraw
2:Line(
3:Horizontal
4:Vertical
5:Tangent(
6:DrawF
7↓Shade(
```

# Let's draw!

- Alles om te tekenen staat in het [DRAW]-menu.
- Teken eens:
  - 1 Een lijn → een driehoek
  - 2 Een cirkel
  - 3 Een smily
  - 4 Wat je wilt!
- Let op hoe snel/handig alle functies zijn.
- Dit hoeft niet in een PRGM: kan gewoon op het hoofdscherm.



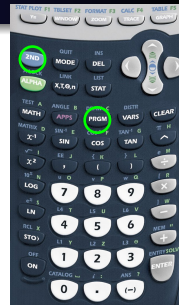
```
:ZStandard
:84→Xmin
:72→Ymax
:ZInteger
```

```
:Line(5,5,35,35)
:Circle(20,20,10)
:DrawF X^1.1
:Shade(X,X²,0,10,1,1)
:Shade(X,X²,10,20,1,2)
:Shade(X,X²,20,30,1,4)
:Shade(X,X²,30,40,2,4)
:Shade(X,X²,40,99,3,7)
```

```
DRAW POINTS STO
1:ClrDraw
2:Line(
3:Horizontal
4:Vertical
5:Tangent(
6:DrawF
7↓Shade(
```

# [DRAW]

- Alles om te tekenen staat in het [DRAW]-menu.
- Een lijn tekenen is vrij snel...
- Maar de Circle $\langle$  functie is zeer langzaam!
  - Ongeschikt voor games!

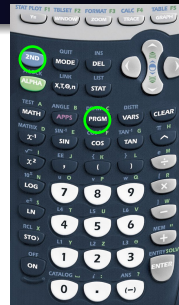


```
DRAW POINTS STO
1:ClrDraw
2:Line(
3:Horizontal
4:Vertical
5:Tangent(
6:DrawF
7:Shade(
```



# [DRAW]

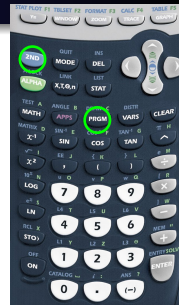
- Alles om te tekenen staat in het [DRAW]-menu.
- Een lijn tekenen is vrij snel...
- Maar de Circle $\langle$  functie is zeer langzaam!
  - Ongeschikt voor games!



```
DRAW POINTS STO
1:ClrDraw
2:Line(
3:Horizontal
4:Vertical
5:Tangent(
6:DrawF
7:Shade(
```

# [DRAW]

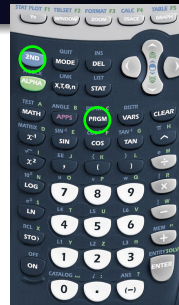
- Alles om te tekenen staat in het [DRAW]-menu.
- Een lijn tekenen is vrij snel...
- Maar de Circle $\langle$  functie is zeer langzaam!
  - Ongeschikt voor games!



```
DRAW POINTS STO
1:ClrDraw
2:Line(
3:Horizontal
4:Vertical
5:Tangent(
6:DrawF
7:Shade(
```

# [DRAW]

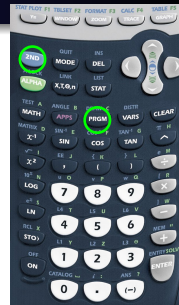
- Alles om te tekenen staat in het [DRAW]-menu.
- Een lijn tekenen is vrij snel...
- Maar de Circle $\langle$  functie is zeer langzaam!
  - Ongeschikt voor games!



```
DRAW POINTS STO
1:ClrDraw
2:Line(
3:Horizontal
4:Vertical
5:Tangent(
6:DrawF
7:Shade(
```

## Per pixel drawing

- Je kunt ook pixels/punten individueel aan en uit zetten.
- Dit bevindt zich onder **POINTS** in het **[DRAW]**-menu.
- Waarom je dit ooit zou doen?
- Maar kost veel geheugen om iets te tekenen...



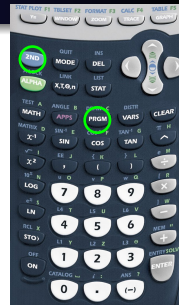
```

0000 POINTS STO
0001 ClrDraw
0002 Line(
0003 Horizontal
0004 Vertical
0005 Tangent(
0006 DrawF
0007 ↓Shade(

```

# Per pixel drawing

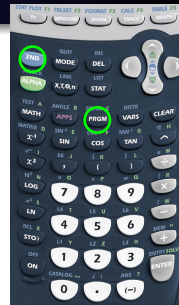
- Je kunt ook pixels/punten individueel aan en uit zetten.
- Dit bevindt zich onder **POINTS** in het [DRAW]-menu.
- Waarom je dit ooit zou doen?
- Maar kost veel geheugen om iets te tekenen...



```
DRAW POINTS STO  
1:Pt-On(  
2:Pt-Off(  
3:Pt-Change(  
4:Pxl-On(  
5:Pxl-Off(  
6:Pxl-Change(  
7:Pxl-Test(  
8:Pt-On(  
9:Pt-Off(  
10:Pt-Change(  
11:Pxl-On(  
12:Pxl-Off(  
13:Pxl-Change(  
14:Pxl-Test(  
15:Pt-On(  
16:Pt-Off(  
17:Pt-Change(  
18:Pxl-On(  
19:Pxl-Off(  
20:Pxl-Change(  
21:Pxl-Test(  
22:Pt-On(  
23:Pt-Off(  
24:Pt-Change(  
25:Pxl-On(  
26:Pxl-Off(  
27:Pxl-Change(  
28:Pxl-Test(  
29:Pt-On(  
30:Pt-Off(  
31:Pt-Change(  
32:Pxl-On(  
33:Pxl-Off(  
34:Pxl-Change(  
35:Pxl-Test(  
36:Pt-On(  
37:Pt-Off(  
38:Pt-Change(  
39:Pxl-On(  
40:Pxl-Off(  
41:Pxl-Change(  
42:Pxl-Test(  
43:Pt-On(  
44:Pt-Off(  
45:Pt-Change(  
46:Pxl-On(  
47:Pxl-Off(  
48:Pxl-Change(  
49:Pxl-Test(  
50:Pt-On(  
51:Pt-Off(  
52:Pt-Change(  
53:Pxl-On(  
54:Pxl-Off(  
55:Pxl-Change(  
56:Pxl-Test(  
57:Pt-On(  
58:Pt-Off(  
59:Pt-Change(  
60:Pxl-On(  
61:Pxl-Off(  
62:Pxl-Change(  
63:Pxl-Test(  
64:Pt-On(  
65:Pt-Off(  
66:Pt-Change(  
67:Pxl-On(  
68:Pxl-Off(  
69:Pxl-Change(  
70:Pxl-Test(  
71:Pt-On(  
72:Pt-Off(  
73:Pt-Change(  
74:Pxl-On(  
75:Pxl-Off(  
76:Pxl-Change(  
77:Pxl-Test(  
78:Pt-On(  
79:Pt-Off(  
80:Pt-Change(  
81:Pxl-On(  
82:Pxl-Off(  
83:Pxl-Change(  
84:Pxl-Test(  
85:Pt-On(  
86:Pt-Off(  
87:Pt-Change(  
88:Pxl-On(  
89:Pxl-Off(  
90:Pxl-Change(  
91:Pxl-Test(  
92:Pt-On(  
93:Pt-Off(  
94:Pt-Change(  
95:Pxl-On(  
96:Pxl-Off(  
97:Pxl-Change(  
98:Pxl-Test(  
99:Pt-On(  
100:Pt-Off(  
101:Pt-Change(  
102:Pxl-On(  
103:Pxl-Off(  
104:Pxl-Change(  
105:Pxl-Test(  
106:Pt-On(  
107:Pt-Off(  
108:Pt-Change(  
109:Pxl-On(  
110:Pxl-Off(  
111:Pxl-Change(  
112:Pxl-Test(  
113:Pt-On(  
114:Pt-Off(  
115:Pt-Change(  
116:Pxl-On(  
117:Pxl-Off(  
118:Pxl-Change(  
119:Pxl-Test(  
120:Pt-On(  
121:Pt-Off(  
122:Pt-Change(  
123:Pxl-On(  
124:Pxl-Off(  
125:Pxl-Change(  
126:Pxl-Test(  
127:Pt-On(  
128:Pt-Off(  
129:Pt-Change(  
130:Pxl-On(  
131:Pxl-Off(  
132:Pxl-Change(  
133:Pxl-Test(  
134:Pt-On(  
135:Pt-Off(  
136:Pt-Change(  
137:Pxl-On(  
138:Pxl-Off(  
139:Pxl-Change(  
140:Pxl-Test(  
141:Pt-On(  
142:Pt-Off(  
143:Pt-Change(  
144:Pxl-On(  
145:Pxl-Off(  
146:Pxl-Change(  
147:Pxl-Test(  
148:Pt-On(  
149:Pt-Off(  
150:Pt-Change(  
151:Pxl-On(  
152:Pxl-Off(  
153:Pxl-Change(  
154:Pxl-Test(  
155:Pt-On(  
156:Pt-Off(  
157:Pt-Change(  
158:Pxl-On(  
159:Pxl-Off(  
160:Pxl-Change(  
161:Pxl-Test(  
162:Pt-On(  
163:Pt-Off(  
164:Pt-Change(  
165:Pxl-On(  
166:Pxl-Off(  
167:Pxl-Change(  
168:Pxl-Test(  
169:Pt-On(  
170:Pt-Off(  
171:Pt-Change(  
172:Pxl-On(  
173:Pxl-Off(  
174:Pxl-Change(  
175:Pxl-Test(  
176:Pt-On(  
177:Pt-Off(  
178:Pt-Change(  
179:Pxl-On(  
180:Pxl-Off(  
181:Pxl-Change(  
182:Pxl-Test(  
183:Pt-On(  
184:Pt-Off(  
185:Pt-Change(  
186:Pxl-On(  
187:Pxl-Off(  
188:Pxl-Change(  
189:Pxl-Test(  
190:Pt-On(  
191:Pt-Off(  
192:Pt-Change(  
193:Pxl-On(  
194:Pxl-Off(  
195:Pxl-Change(  
196:Pxl-Test(  
197:Pt-On(  
198:Pt-Off(  
199:Pt-Change(  
200:Pxl-On(  
201:Pxl-Off(  
202:Pxl-Change(  
203:Pxl-Test(  
204:Pt-On(  
205:Pt-Off(  
206:Pt-Change(  
207:Pxl-On(  
208:Pxl-Off(  
209:Pxl-Change(  
210:Pxl-Test(  
211:Pt-On(  
212:Pt-Off(  
213:Pt-Change(  
214:Pxl-On(  
215:Pxl-Off(  
216:Pxl-Change(  
217:Pxl-Test(  
218:Pt-On(  
219:Pt-Off(  
220:Pt-Change(  
221:Pxl-On(  
222:Pxl-Off(  
223:Pxl-Change(  
224:Pxl-Test(  
225:Pt-On(  
226:Pt-Off(  
227:Pt-Change(  
228:Pxl-On(  
229:Pxl-Off(  
230:Pxl-Change(  
231:Pxl-Test(  
232:Pt-On(  
233:Pt-Off(  
234:Pt-Change(  
235:Pxl-On(  
236:Pxl-Off(  
237:Pxl-Change(  
238:Pxl-Test(  
239:Pt-On(  
240:Pt-Off(  
241:Pt-Change(  
242:Pxl-On(  
243:Pxl-Off(  
244:Pxl-Change(  
245:Pxl-Test(  
246:Pt-On(  
247:Pt-Off(  
248:Pt-Change(  
249:Pxl-On(  
250:Pxl-Off(  
251:Pxl-Change(  
252:Pxl-Test(  
253:Pt-On(  
254:Pt-Off(  
255:Pt-Change(  
256:Pxl-On(  
257:Pxl-Off(  
258:Pxl-Change(  
259:Pxl-Test(  
260:Pt-On(  
261:Pt-Off(  
262:Pt-Change(  
263:Pxl-On(  
264:Pxl-Off(  
265:Pxl-Change(  
266:Pxl-Test(  
267:Pt-On(  
268:Pt-Off(  
269:Pt-Change(  
270:Pxl-On(  
271:Pxl-Off(  
272:Pxl-Change(  
273:Pxl-Test(  
274:Pt-On(  
275:Pt-Off(  
276:Pt-Change(  
277:Pxl-On(  
278:Pxl-Off(  
279:Pxl-Change(  
280:Pxl-Test(  
281:Pt-On(  
282:Pt-Off(  
283:Pt-Change(  
284:Pxl-On(  
285:Pxl-Off(  
286:Pxl-Change(  
287:Pxl-Test(  
288:Pt-On(  
289:Pt-Off(  
290:Pt-Change(  
291:Pxl-On(  
292:Pxl-Off(  
293:Pxl-Change(  
294:Pxl-Test(  
295:Pt-On(  
296:Pt-Off(  
297:Pt-Change(  
298:Pxl-On(  
299:Pxl-Off(  
300:Pxl-Change(  
301:Pxl-Test(  
302:Pt-On(  
303:Pt-Off(  
304:Pt-Change(  
305:Pxl-On(  
306:Pxl-Off(  
307:Pxl-Change(  
308:Pxl-Test(  
309:Pt-On(  
310:Pt-Off(  
311:Pt-Change(  
312:Pxl-On(  
313:Pxl-Off(  
314:Pxl-Change(  
315:Pxl-Test(  
316:Pt-On(  
317:Pt-Off(  
318:Pt-Change(  
319:Pxl-On(  
320:Pxl-Off(  
321:Pxl-Change(  
322:Pxl-Test(  
323:Pt-On(  
324:Pt-Off(  
325:Pt-Change(  
326:Pxl-On(  
327:Pxl-Off(  
328:Pxl-Change(  
329:Pxl-Test(  
330:Pt-On(  
331:Pt-Off(  
332:Pt-Change(  
333:Pxl-On(  
334:Pxl-Off(  
335:Pxl-Change(  
336:Pxl-Test(  
337:Pt-On(  
338:Pt-Off(  
339:Pt-Change(  
340:Pxl-On(  
341:Pxl-Off(  
342:Pxl-Change(  
343:Pxl-Test(  
344:Pt-On(  
345:Pt-Off(  
346:Pt-Change(  
347:Pxl-On(  
348:Pxl-Off(  
349:Pxl-Change(  
350:Pxl-Test(  
351:Pt-On(  
352:Pt-Off(  
353:Pt-Change(  
354:Pxl-On(  
355:Pxl-Off(  
356:Pxl-Change(  
357:Pxl-Test(  
358:Pt-On(  
359:Pt-Off(  
360:Pt-Change(  
361:Pxl-On(  
362:Pxl-Off(  
363:Pxl-Change(  
364:Pxl-Test(  
365:Pt-On(  
366:Pt-Off(  
367:Pt-Change(  
368:Pxl-On(  
369:Pxl-Off(  
370:Pxl-Change(  
371:Pxl-Test(  
372:Pt-On(  
373:Pt-Off(  
374:Pt-Change(  
375:Pxl-On(  
376:Pxl-Off(  
377:Pxl-Change(  
378:Pxl-Test(  
379:Pt-On(  
380:Pt-Off(  
381:Pt-Change(  
382:Pxl-On(  
383:Pxl-Off(  
384:Pxl-Change(  
385:Pxl-Test(  
386:Pt-On(  
387:Pt-Off(  
388:Pt-Change(  
389:Pxl-On(  
390:Pxl-Off(  
391:Pxl-Change(  
392:Pxl-Test(  
393:Pt-On(  
394:Pt-Off(  
395:Pt-Change(  
396:Pxl-On(  
397:Pxl-Off(  
398:Pxl-Change(  
399:Pxl-Test(  
400:Pt-On(  
401:Pt-Off(  
402:Pt-Change(  
403:Pxl-On(  
404:Pxl-Off(  
405:Pxl-Change(  
406:Pxl-Test(  
407:Pt-On(  
408:Pt-Off(  
409:Pt-Change(  
410:Pxl-On(  
411:Pxl-Off(  
412:Pxl-Change(  
413:Pxl-Test(  
414:Pt-On(  
415:Pt-Off(  
416:Pt-Change(  
417:Pxl-On(  
418:Pxl-Off(  
419:Pxl-Change(  
420:Pxl-Test(  
421:Pt-On(  
422:Pt-Off(  
423:Pt-Change(  
424:Pxl-On(  
425:Pxl-Off(  
426:Pxl-Change(  
427:Pxl-Test(  
428:Pt-On(  
429:Pt-Off(  
430:Pt-Change(  
431:Pxl-On(  
432:Pxl-Off(  
433:Pxl-Change(  
434:Pxl-Test(  
435:Pt-On(  
436:Pt-Off(  
437:Pt-Change(  
438:Pxl-On(  
439:Pxl-Off(  
440:Pxl-Change(  
441:Pxl-Test(  
442:Pt-On(  
443:Pt-Off(  
444:Pt-Change(  
445:Pxl-On(  
446:Pxl-Off(  
447:Pxl-Change(  
448:Pxl-Test(  
449:Pt-On(  
450:Pt-Off(  
451:Pt-Change(  
452:Pxl-On(  
453:Pxl-Off(  
454:Pxl-Change(  
455:Pxl-Test(  
456:Pt-On(  
457:Pt-Off(  
458:Pt-Change(  
459:Pxl-On(  
460:Pxl-Off(  
461:Pxl-Change(  
462:Pxl-Test(  
463:Pt-On(  
464:Pt-Off(  
465:Pt-Change(  
466:Pxl-On(  
467:Pxl-Off(  
468:Pxl-Change(  
469:Pxl-Test(  
470:Pt-On(  
471:Pt-Off(  
472:Pt-Change(  
473:Pxl-On(  
474:Pxl-Off(  
475:Pxl-Change(  
476:Pxl-Test(  
477:Pt-On(  
478:Pt-Off(  
479:Pt-Change(  
480:Pxl-On(  
481:Pxl-Off(  
482:Pxl-Change(  
483:Pxl-Test(  
484:Pt-On(  
485:Pt-Off(  
486:Pt-Change(  
487:Pxl-On(  
488:Pxl-Off(  
489:Pxl-Change(  
490:Pxl-Test(  
491:Pt-On(  
492:Pt-Off(  
493:Pt-Change(  
494:Pxl-On(  
495:Pxl-Off(  
496:Pxl-Change(  
497:Pxl-Test(  
498:Pt-On(  
499:Pt-Off(  
500:Pt-Change(  
501:Pxl-On(  
502:Pxl-Off(  
503:Pxl-Change(  
504:Pxl-Test(  
505:Pt-On(  
506:Pt-Off(  
507:Pt-Change(  
508:Pxl-On(  
509:Pxl-Off(  
510:Pxl-Change(  
511:Pxl-Test(  
512:Pt-On(  
513:Pt-Off(  
514:Pt-Change(  
515:Pxl-On(  
516:Pxl-Off(  
517:Pxl-Change(  
518:Pxl-Test(  
519:Pt-On(  
520:Pt-Off(  
521:Pt-Change(  
522:Pxl-On(  
523:Pxl-Off(  
524:Pxl-Change(  
525:Pxl-Test(  
526:Pt-On(  
527:Pt-Off(  
528:Pt-Change(  
529:Pxl-On(  
530:Pxl-Off(  
531:Pxl-Change(  
532:Pxl-Test(  
533:Pt-On(  
534:Pt-Off(  
535:Pt-Change(  
536:Pxl-On(  
537:Pxl-Off(  
538:Pxl-Change(  
539:Pxl-Test(  
540:Pt-On(  
541:Pt-Off(  
542:Pt-Change(  
543:Pxl-On(  
544:Pxl-Off(  
545:Pxl-Change(  
546:Pxl-Test(  
547:Pt-On(  
548:Pt-Off(  
549:Pt-Change(  
550:Pxl-On(  
551:Pxl-Off(  
552:Pxl-Change(  
553:Pxl-Test(  
554:Pt-On(  
555:Pt-Off(  
556:Pt-Change(  
557:Pxl-On(  
558:Pxl-Off(  
559:Pxl-Change(  
560:Pxl-Test(  
561:Pt-On(  
562:Pt-Off(  
563:Pt-Change(  
564:Pxl-On(  
565:Pxl-Off(  
566:Pxl-Change(  
567:Pxl-Test(  
568:Pt-On(  
569:Pt-Off(  
570:Pt-Change(  
571:Pxl-On(  
572:Pxl-Off(  
573:Pxl-Change(  
574:Pxl-Test(  
575:Pt-On(  
576:Pt-Off(  
577:Pt-Change(  
578:Pxl-On(  
579:Pxl-Off(  
580:Pxl-Change(  
581:Pxl-Test(  
582:Pt-On(  
583:Pt-Off(  
584:Pt-Change(  
585:Pxl-On(  
586:Pxl-Off(  
587:Pxl-Change(  
588:Pxl-Test(  
589:Pt-On(  
590:Pt-Off(  
591:Pt-Change(  
592:Pxl-On(  
593:Pxl-Off(  
594:Pxl-Change(  
595:Pxl-Test(  
596:Pt-On(  
597:Pt-Off(  
598:Pt-Change(  
599:Pxl-On(  
600:Pxl-Off(  
601:Pxl-Change(  
602:Pxl-Test(  
603:Pt-On(  
604:Pt-Off(  
605:Pt-Change(  
606:Pxl-On(  
607:Pxl-Off(  
608:Pxl-Change(  
609:Pxl-Test(  
610:Pt-On(  
611:Pt-Off(  
612:Pt-Change(  
613:Pxl-On(  
614:Pxl-Off(  
615:Pxl-Change(  
616:Pxl-Test(  
617:Pt-On(  
618:Pt-Off(  
619:Pt-Change(  
620:Pxl-On(  
621:Pxl-Off(  
622:Pxl-Change(  
623:Pxl-Test(  
624:Pt-On(  
625:Pt-Off(  
626:Pt-Change(  
627:Pxl-On(  
628:Pxl-Off(  
629:Pxl-Change(  
630:Pxl-Test(  
631:Pt-On(  
632:Pt-Off(  
633:Pt-Change(  
634:Pxl-On(  
635:Pxl-Off(  
636:Pxl-Change(  
637:Pxl-Test(  
638:Pt-On(  
639:Pt-Off(  
640:Pt-Change(  
641:Pxl-On(  
642:Pxl-Off(  
643:Pxl-Change(  
644:Pxl-Test(  
645:Pt-On(  
646:Pt-Off(  
647:Pt-Change(  
648:Pxl-On(  
649:Pxl-Off(  
650:Pxl-Change(  
651:Pxl-Test(  
652:Pt-On(  
653:Pt-Off(  
654:Pt-Change(  
655:Pxl-On(  
656:Pxl-Off(  
657:Pxl-Change(  
658:Pxl-Test(  
659:Pt-On(  
660:Pt-Off(  
661:Pt-Change(  
662:Pxl-On(  
663:Pxl-Off(  
664:Pxl-Change(  
665:Pxl-Test(  
666:Pt-On(  
667:Pt-Off(  
668:Pt-Change(  
669:Pxl-On(  
670:Pxl-Off(  
671:Pxl-Change(  
672:Pxl-Test(  
673:Pt-On(  
674:Pt-Off(  
675:Pt-Change(  
676:Pxl-On(  
677:Pxl-Off(  
678:Pxl-Change(  
679:Pxl-Test(  
680:Pt-On(  
681:Pt-Off(  
682:Pt-Change(  
683:Pxl-On(  
684:Pxl-Off(  
685:Pxl-Change(  
686:Pxl-Test(  
687:Pt-On(  
688:Pt-Off(  
689:Pt-Change(  
690:Pxl-On(  
691:Pxl-Off(  
692:Pxl-Change(  
693:Pxl-Test(  
694:Pt-On(  
695:Pt-Off(  
696:Pt-Change(  
697:Pxl-On(  
698:Pxl-Off(  
699:Pxl-Change(  
700:Pxl-Test(  
701:Pt-On(  
702:Pt-Off(  
703:Pt-Change(  
704:Pxl-On(  
705:Pxl-Off(  
706:Pxl-Change(  
707:Pxl-Test(  
708:Pt-On(  
709:Pt-Off(  
710:Pt-Change(  
711:Pxl-On(  
712:Pxl-Off(  
713:Pxl-Change(  
714:Pxl-Test(  
715:Pt-On(  
716:Pt-Off(  
717:Pt-Change(  
718:Pxl-On(  
719:Pxl-Off(  
720:Pxl-Change(  
721:Pxl-Test(  
722:Pt-On(  
723:Pt-Off(  
724:Pt-Change(  
725:Pxl-On(  
726:Pxl-Off(  
727:Pxl-Change(  
728:Pxl-Test(  
729:Pt-On(  
730:Pt-Off(  
731:Pt-Change(  
732:Pxl-On(  
733:Pxl-Off(  
734:Pxl-Change(  
735:Pxl-Test(  
736:Pt-On(  
737:Pt-Off(  
738:Pt-Change(  
739:Pxl-On(  
740:Pxl-Off(  
741:Pxl-Change(  
742:Pxl-Test(  
743:Pt-On(  
744:Pt-Off(  
745:Pt-Change(  
746:Pxl-On(  
747:Pxl-Off(  
748:Pxl-Change(  
749:Pxl-Test(  
750:Pt-On(  
751:Pt-Off(  
752:Pt-Change(  
753:Pxl-On(  
754:Pxl-Off(  
755:Pxl-Change(  
756:Pxl-Test(  
757:Pt-On(  
758:Pt-Off(  
759:Pt-Change(  
760:Pxl-On(  
761:Pxl-Off(  
762:Pxl-Change(  
763:Pxl-Test(  
764:Pt-On(  
765:Pt-Off(  
766:Pt-Change(  
767:Pxl-On(  
768:Pxl-Off(  
769:Pxl-Change(  
770:Pxl-Test(  
771:Pt-On(  
772:Pt-Off(  
773:Pt-Change(  
774:Pxl-On(  
775:Pxl-Off(  
776:Pxl-Change(  
777:Pxl-Test(  
778:Pt-On(  
779:Pt-Off(  
780:Pt-Change(  
781:Pxl-On(  
782:Pxl-Off(  
783:Pxl-Change(  
784:Pxl-Test(  
785:Pt-On(  
786:Pt-Off(  
787:Pt-Change(  
788:Pxl-On(  
789:Pxl-Off(  
790:Pxl-Change(  
791:Pxl-Test(  
792:Pt-On(  
793:Pt-Off(  
794:Pt-Change(  
795:Pxl-On(  
796:Pxl-Off(  
797:Pxl-Change(  
798:Pxl-Test(  
799:Pt-On(  
800:Pt-Off(  
801:Pt-Change(  
802:Pxl-On(  
803:Pxl-Off(  
804:Pxl-Change(  
805:Pxl-Test(  
806:Pt-On(  
807:Pt-Off(  
808:Pt-Change(  
809:Pxl-On(  
810:Pxl-Off(  
811:Pxl-Change(  
812:Pxl-Test(  
813:Pt-On(  
814:Pt-Off(  
815:Pt-Change(  
816:Pxl-On(  
817:Pxl-Off(  
818:Pxl-Change(  
819:Pxl-Test(  
820:Pt-On(  
821:Pt-Off(  
822:Pt-Change(  
823:Pxl-On(  
824:Pxl-Off(  
825:Pxl-Change(  
826:Pxl-Test(  
827:Pt-On(  
828:Pt-Off(  
829:Pt-Change(  
830:Pxl-On(  
831:Pxl-Off(  
832:Pxl-Change(  
833:Pxl-Test(  
834:Pt-On(  
835:Pt-Off(  
836:Pt-Change(  
837:Pxl-On(  
838:Pxl-Off(  
839:Pxl-Change(  
840:Pxl-Test(  
841:Pt-On(  
842:Pt-Off(  
843:Pt-Change(  
844:Pxl-On(  
845:Pxl-Off(  
846:Pxl-Change(  
847:Pxl-Test(  
848:Pt-On(  
849:Pt-Off(  
850:Pt-Change(  
851:Pxl-On(  
852:Pxl-Off(  
853:Pxl-Change(  
854:Pxl-Test(  
855:Pt-On(  
856:Pt-Off(  
857:Pt-Change(  
858:Pxl-On(  
859:Pxl-Off(  
860:Pxl-Change(  
861:Pxl-Test(  
862:Pt-On(  
863:Pt-Off(  
864:Pt-Change(  
865:Pxl-On(  
866:Pxl-Off(  
867:Pxl-Change(  
868:Pxl-Test(  
869:Pt-On(  
870:Pt-Off(  
871:Pt-Change(  
872:Pxl-On(  
873:Pxl-Off(  
874:Pxl-Change(  
875:Pxl-Test(  
876:Pt-On(  
877:Pt-Off(  
878:Pt-Change(  
879:Pxl-On(  
880:Pxl-Off(  
881:Pxl-Change(  
882:Pxl-Test(  
883:Pt-On(  
884:Pt-Off(  
885:Pt-Change(  
886:Pxl-On(  
887:Pxl-Off(  
888:Pxl-Change(  
889:Pxl-Test(  
890:Pt-On(  
891:Pt-Off(  
892:Pt-Change(  
893:Pxl-On(  
894:Pxl-Off(  
895:Pxl-Change(  
896:Pxl-Test(  
897:Pt-On(  
898:Pt-Off(  
899:Pt-Change(  
900:Pxl-On(  
901:Pxl-Off(  
902:Pxl-Change(  
903:Pxl-Test(  
904:Pt-On(  
905:Pt-Off(  
906:Pt-Change(  
907:Pxl-On(  
908:Pxl-Off(  
909:Pxl-Change(  
910:Pxl-Test(  
911:Pt-On(  
912:Pt-Off(  
913:Pt-Change(  
914:Pxl-On(  
915:Pxl-Off(  
916:Pxl-Change(  
917:Pxl-Test(  
918:Pt-On(  
919:Pt-Off(  
920:Pt-Change(  
921:Pxl-On(  
922:Pxl-Off(  
923:Pxl-Change(  
924:Pxl-Test(  
925:Pt-On(  
926:Pt-Off(  
927:Pt-Change(  
928:Pxl-On(  
929:Pxl-Off(  
930:Pxl-Change(  
931:Pxl-Test(  
932:Pt-On(  
933:Pt-Off(  
934:Pt-Change(  
935:Pxl-On(  
936:Pxl-Off(  
937:Pxl-Change(  
938:Pxl-Test(  
939:Pt-On(  
940:Pt-Off(  
941:Pt-Change(  
942:Pxl-On(  
943:Pxl-Off(  
944:Pxl-Change(  
945:Pxl-Test(  
946:Pt-On(  
947:Pt-Off(  
948:Pt-Change(  
949:Pxl-On(  
950:Pxl-Off(  
951:Pxl-Change(  
952:Pxl-Test(  
953:Pt-On(  
954:Pt-Off(  
955:Pt-Change(  
956:Pxl-On(  
957:Pxl-Off(  
958:Pxl-Change(  
959:Pxl-Test(  
960:Pt-On(  
961:Pt-Off(  
962:Pt-Change(  
963:Pxl-On(  
964:Pxl-Off(  
965:Pxl-Change(  
966:Pxl-Test(  
967:Pt-On(  
968:Pt-Off(  
969:Pt-Change(  
970:Pxl-On(  
971:Pxl-Off(  
972:Pxl-Change(  
973:Pxl-Test(  
974:Pt-On(  
975:Pt-Off(  
976:Pt-Change(  
977:Pxl-On(  
978:Pxl-Off(  
979:Pxl-Change(  
980:Pxl-Test(  
981:Pt-On(  
982:Pt-Off(  
983:Pt-Change(  
984:Pxl-On(  
985:Pxl-Off(  
986:Pxl-Change(  
987:Pxl-Test(  
988:Pt-On(  
989:Pt-Off(  
990:Pt-Change(  
991:Pxl-On(  
992:Pxl-Off(  
993:Pxl-Change(  
994:Pxl-Test(  
995:Pt-On(  
996:Pt-Off(  
997:Pt-Change(  
998:Pxl-On(  
999:Pxl-Off(  
1000:Pxl-Change(  
1001:Pxl-Test(  
1002:Pt-On(  
1003:Pt-Off(  
1004:Pt-Change(  
1005:Pxl-On(  
1006:Pxl-Off(  
1007:Pxl-Change(  
1008:Pxl-Test(  
1009:Pt-On(  
1010:Pt-Off(  
1011:Pt-Change(  
1012:Pxl-On(  
1013:Pxl-Off(  
1014:Pxl-Change(  
1015:Pxl-Test(  
1016:Pt-On(  
1017:Pt-Off(  
1018:Pt-Change(  
1019:Pxl-On(  
1020:Pxl-Off(  
1021:Pxl-Change(  
1022:Pxl-Test(  
1023:Pt-On(  
1024:Pt-Off(  
1025:Pt-Change(  
1026:Pxl-On(  
1027:Pxl-Off(  
1028:Pxl-Change(  
1029:Pxl-Test(  
1030:Pt-On(  
1031:Pt-Off(  
1032:Pt-Change(  
1033:Pxl-On(  
1034:Pxl-Off(  
1035:Pxl-Change(  
1036:Pxl-Test(  
1037:Pt-On(  
1038:Pt-Off(  
1039:Pt-Change(  
1040:Pxl-On(  
1041:Pxl-Off(  
1042:Pxl-Change(  
1043:Pxl-Test(  
1044:Pt-On(  
1045:Pt-Off(  
1046:Pt-Change(  
1047:Pxl-On(  
1048:Pxl-Off(  
1049:Pxl-Change(  
1050:Pxl-Test(  
1051:Pt-On(  
1052:Pt-Off(  
1053:Pt-Change(  
1054:Pxl-On(  
1055:Pxl-Off(  
1056:Pxl-Change(  
1057:Pxl-Test(  
1058:Pt-On(  
1059:Pt-Off(  
1060:Pt-Change(  
1061:Pxl-On(  
1062:Pxl-Off(  
1063:Pxl-Change(  
1064:Pxl-Test(  
1065:Pt-On(  
1066:Pt-Off(  
1067:Pt-Change(  
1068:Pxl-On(  
1069:Pxl-Off(  
1070:Pxl-Change(  
1071:Pxl-Test(  
1072:Pt-On(  
1073:Pt-Off(  
1074:Pt-Change(  
1075:Pxl-On(  
1076:Pxl-Off(  
1077:Pxl-Change(  
1078:Pxl-Test(  
1079:Pt-On(  
1080:Pt-Off(  
1081:Pt-Change(  
1082:Pxl-On(  
1083:Pxl-Off(  
1084:Pxl-Change(  
1085:Pxl-Test(  
1086:Pt-On(  
1087:Pt-Off(  
1088:Pt-Change(  
1089:Pxl-On(  
1090:Pxl-Off(  
1091:Pxl-Change(  
1092:Pxl-Test(  
1093:Pt-On(  
1094:Pt-Off(  
1095:Pt-Change(  
1096:Pxl-On(  
1097:Pxl-Off(  
1098:Pxl-Change(  
1099:Pxl-Test(  
1100:Pt-On(  
1101:Pt-Off(  
1102:Pt-Change(  
1103:Pxl-On(  
1104:Pxl-Off(  
1105:Pxl-Change(  
1106:Pxl-Test(  
1107:Pt-On(  
1108:Pt-Off(  
1109:Pt-Change(  
1110:Pxl-On(  
1111:Pxl-Off(  
1112:Pxl-Change(  
1113:Pxl-Test(  
1114:Pt-On(  
1115:Pt-Off(  
1116:Pt-Change(  
1117:Pxl-On(  
1118:Pxl-Off(  
1119:Pxl-Change(  
1120:Pxl-Test(  
1121:Pt-On(  
1122:Pt-Off(  
1123:Pt-Change(  
1124:Pxl-On(  
1125:Pxl-Off(  
1126:Pxl-Change(  
1127:Pxl-Test(  
1128:Pt-On(  
1129:Pt-Off(  
1130:Pt-Change(  
1131:Pxl-On(  
1132:Pxl-Off(  
1133:Pxl-Change(  
1134:Pxl-Test(  
1135:Pt-On(  
1136:Pt-Off(  
1137:Pt-Change(  
1138:Pxl-On(  
1139:Pxl-Off(  
1140:Pxl-Change(  
1141:Pxl-Test(  
1142:Pt-On(  
1143:Pt-Off(  
1144:Pt-Change(  
1145:Pxl-On(  
1146:Pxl-Off(  
1147:Pxl-Change(  
1148:Pxl-Test(  
1149:Pt-On(  
1150:Pt-Off(  
1151:Pt-Change(  
1152:Pxl-On(  
1153:Pxl-Off(  
1154:Pxl-Change(  
1155:Pxl-Test(  
1156:Pt-On(  
1157:Pt-Off(  
1158:Pt-Change(  
1159:Pxl-On(  
1160:Pxl-Off(  
1161:Pxl-Change(  
1162:Pxl-Test(  
1163:Pt-On(  
1164:Pt-Off(  
1165:Pt-Change(  
1166:Pxl-On(  
1167:Pxl-Off(  
1168:Pxl-Change(  
1169:Pxl-Test(  
1170:Pt-On(  
1171:Pt-Off(  
1172:Pt-Change(  
1173:Pxl-On(  
1174:Pxl-Off(  
1175:Pxl-Change(  
1176:Pxl-Test(  
1177:Pt-On(  
1178:Pt-Off(  
1179:Pt-Change(  
1180:Pxl-On(  
1181:Pxl-Off(  
1182:Pxl-Change(  
1183:Pxl-Test(  
1184:Pt-On(  
1185:Pt-Off(  
1186:Pt-Change(  
1187:Pxl-On(  
1188:Pxl-Off(  
1189:Pxl-Change(  
1190:Pxl-Test(  
1191:Pt-On(  
1192:Pt-Off(  
1193:Pt-Change(  
1194:Pxl-On(  
1195:Pxl-Off(  
1196:Pxl-Change(  
1197:Pxl-Test(  
1198:Pt-On(  
1199:Pt-Off(  
1200:Pt-Change(  
1201:Pxl-On(  
1202:Pxl-Off(  
1203:Pxl-Change(  
1204:Pxl-Test(  
1205:Pt-On(  
1206:Pt-Off(  
1207:Pt-Change(  
1208:Pxl-On(  
1209:Pxl-Off(  
1210:Pxl-Change(  
1211:Pxl-Test(  
1212:Pt-On(  
1213:Pt-Off(  
1214:Pt-Change(  
1215:Pxl-On(  
1216:Pxl-Off(  
1217:Pxl-Change(  
1218:Pxl-Test(  
1219:Pt-On(  
1220:Pt-Off(  
1221:Pt-Change(  
1222:Pxl-On(  
1223:Pxl-Off(  
1224:Pxl-Change(  
12
```

# Per pixel drawing

- Je kunt ook pixels/punten individueel aan en uit zetten.
- Dit bevindt zich onder **POINTS** in het [DRAW]-menu.
- Waarom je dit ooit zou doen?



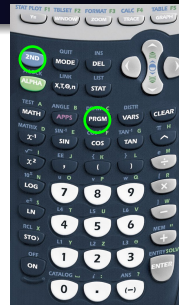
- Maar kost veel geheugen om iets te tekenen...

```

DRAW POINTS STO
1:Pt-On(
2:Pt-Off(
3:Pt-Change(
4:Pxl-On(
5:Pxl-Off(
6:Pxl-Change(
7:Pxl-Test(
  
```

# Per pixel drawing

- Je kunt ook pixels/punten individueel aan en uit zetten.
- Dit bevindt zich onder **POINTS** in het [DRAW]-menu.
- Waarom je dit ooit zou doen?
  - Veel sneller!! Je wilt een real-time game!
- Maar kost veel geheugen om iets te tekenen...

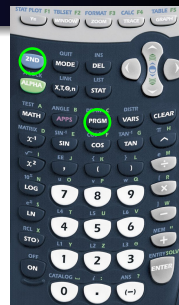


```

DRAW POINTS STO
1:Pt-On(
2:Pt-Off(
3:Pt-Change(
4:Pxl-On(
5:Pxl-Off(
6:Pxl-Change(
7:Pxl-Test(
  
```

# Per pixel drawing

- Je kunt ook pixels/punten individueel aan en uit zetten.
- Dit bevindt zich onder **POINTS** in het [DRAW]-menu.
- Waarom je dit ooit zou doen?
  - Veel sneller!! Je wilt een real-time game!
- Maar kost veel geheugen om iets te tekenen...



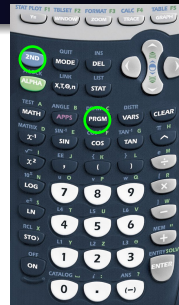
```

DRAW POINTS STO
1:Pt-On(
2:Pt-Off(
3:Pt-Change(
4:Pxl-On(
5:Pxl-Off(
6:Pxl-Change(
7:Pxl-Test(
  
```



# Per pixel drawing

- Punten (x,y) beginnen linksonderin.
- Pixels (R,C) beginnen linksbovenin.
  - R (rij) telt verticaal van boven naar beneden!
  - C (kolom) telt horizontaal
- Bij welk punt hoort pixel (R,C)? (Tip:  $y_{max} = 62$ )
- En bij welke pixel hoort punt (x,y)?



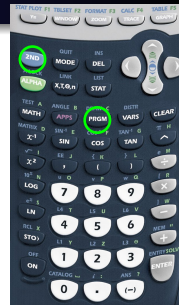
```

DRAW POINTS STO
1:Pt-On(
2:Pt-Off(
3:Pt-Change(
4:Pxl-On(
5:Pxl-Off(
6:Pxl-Change(
7:Pxl-Test(

```

## Per pixel drawing

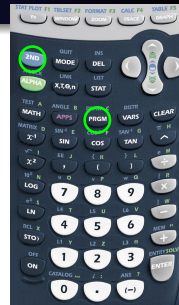
- Punten (x,y) beginnen linksonderin.
- Pixels (R,C) beginnen linksbovenin.
  - R (rij) telt verticaal van boven naar beneden!
  - C (kolom) telt horizontaal
- Bij welk punt hoort pixel (R,C)? (Tip:  $y_{max} = 62$ )
- En bij welke pixel hoort punt (x,y)?



```
DRAW POINTS STO
1:Pt-On(
2:Pt-Off(
3:Pt-Change(
4:Pxl-On(
5:Pxl-Off(
6:Pxl-Change(
7:pxl-Test(
```

## Per pixel drawing

- Punten (x,y) beginnen linksonderin.
- Pixels (R,C) beginnen linksbovenin.
  - R (rij) telt verticaal van boven naar beneden!
  - C (kolom) telt horizontaal
- Bij welk punt hoort pixel (R,C)? (Tip:  $y_{max} = 62$ )



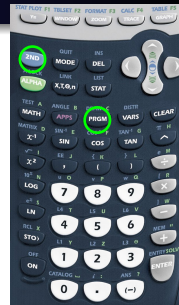
```

DRAW POINTS STO
1:Pt-On(
2:Pt-Off(
3:Pt-Change(
4:Pxl-On(
5:Pxl-Off(
6:Pxl-Change(
7:pxl-Test(

```

## Per pixel drawing

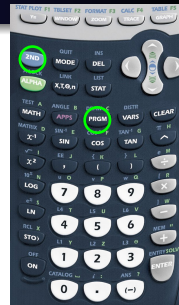
- Punten (x,y) beginnen linksonderin.
- Pixels (R,C) beginnen linksbovenin.
  - R (rij) telt verticaal van boven naar beneden!
  - C (kolom) telt horizontaal
- Bij welk punt hoort pixel (R,C)? (Tip:  $y_{max} = 62$ )
- En bij welke pixel hoort punt (x,y)?



```
DRAW POINTS STO  
1:Pt-On(  
2:Pt-Off(  
3:Pt-Change(  
4:Pxl-On(  
5:Pxl-Off(  
6:Pxl-Change(  
7:pxl-Test(  
8:END
```

## Per pixel drawing

- Punten  $(x,y)$  beginnen linksonderin.
- Pixels  $(R,C)$  beginnen linksbovenin.
  - $R$  (rij) telt verticaal van boven naar beneden!
  - $C$  (kolom) telt horizontaal
- Bij welk punt hoort pixel  $(R,C)$ ? (Tip:  $y_{max} = 62$ )
  - Punt  $(C, 62-R)$
- En bij welke pixel hoort punt  $(x,y)$ ?
  - Pixel  $(62-y, x)$



```
DRAW POINTS STO
1:Pt-On(
2:Pt-Off(
3:Pt-Change(
4:Pxl-On(
5:Pxl-Off(
6:Pxl-Change(
7:pxl-Test(
```

# Outline

- 1 Graphics
  - Friendly window
  - Basic drawing functions
  - Sprites
- 2 Dynamic input
  - getKey
  - wait for input
  - Memory Leaks
- 3 Games!
  - Game Loop
  - Towards multiplayer...
- 4 Exercises
  - Exercises
  - Further Reading

## Sprite: Meerdere lijnen

- In een game wil je vaak een object tekenen, zoals pacman.
- Zo'n object heet een “sprite”.
- Op de TI84 is het een combinatie van meerdere pixels of lijnen.

## Sprite: Meerdere lijnen

- In een game wil je vaak een object tekenen, zoals pacman.
- Zo'n object heet een “sprite”.
- Op de TI84 is het een combinatie van meerdere pixels of lijnen.



## Sprite: Meerdere lijnen

- In een game wil je vaak een object tekenen, zoals pacman.
- Zo'n object heet een “sprite”.
- Op de TI84 is het een combinatie van meerdere pixels of lijnen.

## Sprite: Meerdere lijnen

- In een game wil je vaak een object tekenen, zoals pacman.
- Zo'n object heet een "sprite".
- Op de TI84 is het een combinatie van meerdere pixels of lijnen.

```
PROGRAM: ØFWINDOW
:FnOff
:GridOff
:AxesOff
:ZStandard
:84→Xmin
:72→Ymax
:ZInteger
:ClrDraw
```

```
PROGRAM: SPRITE
:PrgrmØFWINDOW
:Line(10,10,20,20
:Line(20,20,30,10
:Line(30,10,20,15
:Line(20,15,10,10
```

# Plot Sprites

- Hoe kan dit efficiënter?

```
PROGRAM: SPRITE  
:PRGM0FWINDOW  
:Line(10,10,20,20  
:Line(20,20,30,10  
:Line(30,10,20,15  
:Line(20,15,10,10
```

# Plot Sprites

- Hoe kan dit efficiënter?
- Store alle getallen in een list!
- Met “Plot” kun je **maximaal 3** sprites maken.
- Voordeel: Sprite makkelijk te bewerken en bewegen.
- Nadeel: `DisFGraph` overschrijft het hele scherm.

```
PROGRAM: SPRITEPL  
:PRGM0FWINDOW  
:(10,20,30,20,10→L1  
:(10,20,10,15,10→L2
```

```
PROGRAM: SPRITE  
:PRGM0FWINDOW  
:Line(10,10,20,20  
:Line(20,20,30,10  
:Line(30,10,20,15  
:Line(20,15,10,10
```

# Plot Sprites

- Hoe kan dit efficiënter?
- Store alle getallen in een list!
- Met “Plot” kun je **maximaal 3** sprites maken.
- Voordeel: Sprite makkelijk te bewerken en bewegen.
- Nadeel: `DisfGraph` overschrijft het hele scherm.

```
PROGRAM: SPRITEPL  
:PRGM0FWINDOW  
:(10,20,30,20,10→L1  
:(10,20,10,15,10→L2  
:Plot1(xyLine,L1,L2  
:DisfGraph
```

```
PROGRAM: SPRITE  
:PRGM0FWINDOW  
:Line(10,10,20,20  
:Line(20,20,30,10  
:Line(30,10,20,15  
:Line(20,15,10,10
```

# Plot Sprites

- Hoe kan dit efficiënter?
- Store alle getallen in een list!
- Met “Plot” kun je **maximaal 3** sprites maken.
- Voordeel: Sprite makkelijk te bewerken en bewegen.
- Nadeel: `DisfGraph` overschrijft het hele scherm.

```
"TRANSLEER X:  
L1+5→L1  
"ROTEER:  
L1cos(θ)-L2sin(θ)→L3  
L1sin(θ)+L2cos(θ)→L2  
L3→L1
```

```
PROGRAM: SPRITEPL  
:PRGMθFWINDOW  
: (10,20,30,20,10→L1  
: (10,20,10,15,10→L2  
: Plot1(xyLine,L1,L2  
: DisfGraph
```

```
PROGRAM: SPRITE  
:PRGMθFWINDOW  
: Line(10,10,20,20  
: Line(20,20,30,10  
: Line(30,10,20,15  
: Line(20,15,10,10
```

# Plot Sprites

- Hoe kan dit efficiënter?
- Store alle getallen in een list!
- Met “Plot” kun je **maximaal 3** sprites maken.
- Voordeel: Sprite makkelijk te bewerken en bewegen.
- Nadeel: `DispGraph` overschrijft het hele scherm.

```
"TRANSLEER X:  
L1+5→L1  
"ROTEER:  
L1cos(θ)-L2sin(θ)→L3  
L1sin(θ)+L2cos(θ)→L2  
L3→L1
```

```
PROGRAM: SPRITEPL  
:PRGMθFWINDOW  
: (10,20,30,20,10→L1  
: (10,20,10,15,10→L2  
: Plot1(xyLine,L1,L2  
: DispGraph
```

```
PROGRAM: SPRITE  
:PRGMθFWINDOW  
: Line(10,10,20,20  
: Line(20,20,30,10  
: Line(30,10,20,15  
: Line(20,15,10,10
```

# Looping Line Sprites

- We kunnen ook zelf met `Line` over een list lopen!
- Voordeel: Cleared het scherm niet, unlike `DispGraph`.
- Nadeel: Tekent langzamer...
- Voor kleine sprites kun je ook met pixels tekenen.

```
PROGRAM: SPRITE
:PRGM0FWINDOW
:Line(10,10,20,20
:Line(20,20,30,10
:Line(30,10,20,15
:Line(20,15,10,10
```



# Looping Line Sprites

- We kunnen ook zelf met **Line** over een list lopen!
- Voordeel: Cleared het scherm niet, unlike **DispGraph**.
- Nadeel: Tekent langzamer...
- Voor kleine sprites kun je ook met pixels tekenen.

```
PROGRAM: SPRITELL
: prgm0FWINDOW
: (10,20,30,20,10)→L1
: (10,20,10,15,10)→L2
: For(X,1,dim(L1)-1
: Line(L1(X),L2(X),
: L1(X+1),L2(X+1)
: End
```

```
PROGRAM: SPRITE
: prgm0FWINDOW
: Line(10,10,20,20
: Line(20,20,30,10
: Line(30,10,20,15
: Line(20,15,10,10
```

# Looping Line Sprites

- We kunnen ook zelf met **Line** over een list lopen!
- Voordeel: Cleared het scherm niet, unlike **DispGraph**.
- Nadeel: Tekent langzamer...
- Voor kleine sprites kun je ook met pixels tekenen.

```
PROGRAM: SPRITELL
: prgm0FWINDOW
: (10,20,30,20,10)→L1
: (10,20,10,15,10)→L2
: For(X,1,dim(L1)-1
: Line(L1(X),L2(X),
: L1(X+1),L2(X+1)
: End
```

```
PROGRAM: SPRITE
: prgm0FWINDOW
: Line(10,10,20,20
: Line(20,20,30,10
: Line(30,10,20,15
: Line(20,15,10,10
```

# Looping Line Sprites

- We kunnen ook zelf met **Line** over een list lopen!
- Voordeel: Cleared het scherm niet, unlike **DispGraph**.
- Nadeel: Tekent langzamer...
- Voor kleine sprites kun je ook met pixels tekenen.

```
PROGRAM: SPRITELP
:Prgrm0FWINDOW
: (10,20,30,20)→L1
: (10,20,10,15)→L2
: For(X,1,dim(L1)-1
: Line(L1(X),L2(X)
: Pt-On(L1(X),L2(X)
: End
```

```
PROGRAM: SPRITELL
:Prgrm0FWINDOW
: (10,20,30,20,10)→L1
: (10,20,10,15,10)→L2
: For(X,1,dim(L1)-1
: Line(L1(X),L2(X),
: L1(X+1),L2(X+1)
: End
```

```
PROGRAM: SPRITE
:Prgrm0FWINDOW
: Line(10,10,20,20
: Line(20,20,30,10
: Line(30,10,20,15
: Line(20,15,10,10
```

# Compressed Lines

- In al het voorgaande werd een sprite getekend met aaneengesloten lijnen.
- Alternatief: compressie van een lijn in een enkel getal.
- Voordeel: flexibel en slechts 1 list.
- Nadeel: lastiger om mee te werken.

```
PROGRAM: SPRITE
:prgmθFWINDOW
:Line(10,10,20,20
:Line(20,20,30,10
:Line(30,10,20,15
:Line(20,15,10,10
```

```
PROGRAM: SPRITELL
:prgmθFWINDOW
:(10,20,30,20,10→L1
:(10,20,10,15,10→L2
:For(X,1,dim(L1)-1
:Line(L1(X),L2(X),
L1(X+1),L2(X+1
:End
```

# Compressed Lines

- In al het voorgaande werd een sprite getekend met aaneengesloten lijnen.
- Alternatief: compressie van een lijn in een enkel getal.
- Voordeel: flexibel en slechts 1 list.
- Nadeel: lastiger om mee te werken.

```
PROGRAM: SPRITE
:prgmθFWINDOW
:Line(10,10,20,20
:Line(20,20,30,10
:Line(30,10,20,15
:Line(20,15,10,10
```

```
PROGRAM: SPRITELL
:prgmθFWINDOW
:(10,20,30,20,10→L1
:(10,20,10,15,10→L2
:For(X,1,dim(L1)-1
:Line(L1(X),L2(X),
L1(X+1),L2(X+1
:End
```

# Compressed Lines

- In al het voorgaande werd een sprite getekend met aaneengesloten lijnen.
- Alternatief: compressie van een lijn in een enkel getal.
- Voordeel: flexibel en slechts 1 list.
- Nadeel: lastiger om mee te werken.

```
PROGRAM: SPRITE
:Prgrm0FWINDOW
:Line(10,10,20,20
:Line(20,20,30,10
:Line(30,10,20,15
:Line(20,15,10,10
```

```
PROGRAM: SPRITELL
:Prgrm0FWINDOW
:(10,20,30,20,10→L1
:(10,20,10,15,10→L2
:For(X,1,dim(L1)-1
:Line(L1(X),L2(X),
L1(X+1),L2(X+1)
:End
```

```
PROGRAM: SPRITES1
:Prgrm0FWINDOW
:(10102020,20203010
,30102015,20151010
:For(X,1,dim(Ans
:Line(iPart(Ans(X)
/E6),iPart(E2fPart(
Ans(X)/E6)),iPart(E2
fPart(Ans(X)/E4)),E2
fPart(Ans(X)/E2
:End
```

# Compressed Lines

- In al het voorgaande werd een sprite getekend met aaneengesloten lijnen.
- Alternatief: compressie van een lijn in een enkel getal.
- Voordeel: flexibel en slechts 1 list.
- Nadeel: lastiger om mee te werken.

```
PROGRAM: SPRITE
:Prgrm0FWINDOW
:Line(10,10,20,20
:Line(20,20,30,10
:Line(30,10,20,15
:Line(20,15,10,10
```

```
PROGRAM: SPRITELL
:Prgrm0FWINDOW
:(10,20,30,20,10→L1
:(10,20,10,15,10→L2
:For(X,1,dim(L1)-1
:Line(L1(X),L2(X),
L1(X+1),L2(X+1)
:End
```

```
PROGRAM: SPRITES1
:Prgrm0FWINDOW
:(10102020,20203010
,30102015,20151010
:For(X,1,dim(Ans
:Line(iPart(Ans(X)
/E6),iPart(E2fPart(
Ans(X)/E6)),iPart(E2
fPart(Ans(X)/E4)),E2
fPart(Ans(X)/E2
:End
```

# Compressed Lines

- Dezelfde **For**-loop kan gebruikt worden voor *elke* sprite!

```
PROGRAM: SPRITES1
:Prgrm0FWINDOW
: (10102020,20203010
,30102015,20151010
:For(X,1,dim(Ans
:Line(iPart(Ans(X)
/E6),iPart(E2fPart(
Ans(X)/E6)),iPart(E2
fPart(Ans(X)/E4)),E2
fPart(Ans(X)/E2
:End
```



# Compressed Lines

- Dezelfde For-loop kan gebruikt worden voor *elke* sprite!

```
PROGRAM: ØDRAWSP
:For(X,1,dim(Ans
:Line(iPart(Ans(X)
/ E6),iPart(E2fPart(
Ans(X)/E6)),iPart(E2
fPart(Ans(X)/E4)),E2
fPart(Ans(X)/E2
:End
```

```
PROGRAM: SPRITES1b
:PrgrmØFWINDOW
:(10102020,20203010
,30102015,20151010
:PrgrmØDRAWSP
```

```
PROGRAM: SPRITES1a
:PrgrmØFWINDOW
:(10102020,20203010
,30102015,20151010
:For(X,1,dim(Ans
:Line(iPart(Ans(X)
/ E6),iPart(E2fPart(
Ans(X)/E6)),iPart(E2
fPart(Ans(X)/E4)),E2
fPart(Ans(X)/E2
:End
```

# Animation 1

- Een animatie is niets anders dan steeds een andere sprite tekenen (of andere positie/rotatie/...).
- Wat doet deze animatie?

```
PROGRAM: ØDRAWSPT
:For(X,1,dim(Ans
:Line(iPart(Ans(X)
/ E6),iPart(E2fPart(
Ans(X)/E6)),iPart(E2
fPart(Ans(X)/E4)),E2
fPart(Ans(X)/E2
:End
```

# Animation 1

- Een animatie is niets anders dan steeds een andere sprite tekenen (of andere positie/rotatie/...).
- Wat doet deze animatie?

```
PROGRAM: ØDRAWSP
: For(X, 1, dim(Ans
: Line(iPart(Ans(X)
/ E6), iPart(E2 fPart(
Ans(X)/E6)), iPart(E2
fPart(Ans(X)/E4)), E2
fPart(Ans(X)/E2
: End
```

```
PROGRAM: SPRITES2
: prgm ØFWINDOW
: (10102020, 20203010
, 30102015, 20151010) → L1
: For(I, 1, 10
: "UPDATE
: L1+2(E4+1) → L1
: "DRAW
: L1: prgm ØDRAWSP
: End
```

# Animation 1

- Een animatie is niets anders dan steeds een andere sprite tekenen (of andere positie/rotatie/...).
- Wat doet deze animatie?
- PROBLEEM: Je ziet alle vorige sprites ook...

```
PROGRAM: ØDRAWSP
: For(X, 1, dim(Ans
: Line(iPart(Ans(X)
/ E6), iPart(E2 fPart(
Ans(X)/E6)), iPart(E2
fPart(Ans(X)/E4)), E2
fPart(Ans(X)/E2
: End
```

```
PROGRAM: SPRITES2
: prgm ØFWINDOW
: (10102020, 20203010
, 30102015, 20151010) → L1
: For(I, 1, 10
: "UPDATE
: L1+2(E4+1) → L1
: "DRAW
: L1: prgm ØDRAWSP
: End
```

## Animation 2

- OPLOSSING: Gum in een lijn
- PROBLEEM: Overschrijft de achtergrond

```
PROGRAM: ØDRAWSPT
: For(X, 1, dim(Ans
: Line(iPart(Ans(X)
/ E6), iPart(E2 fPart(
Ans(X)/E6)), iPart(E2
fPart(Ans(X)/E4)), E2
fPart(Ans(X)/E2), B
: End
```

```
PROGRAM: SPRITES2
: prgm ØFWINDOW
: (10102020, 20203010
, 30102015, 20151010) → L1
: For(I, 1, 10
: "UPDATE
: L1+2(E4+1) → L1
: "DRAW
: L1: prgm ØDRAWSPT
: End
```

## Animation 2

- OPLOSSING: Gum in een lijn
- PROBLEEM: Overschrijft de achtergrond

```
PROGRAM: ØDRAWSPT
:For(X,1,dim(Ans
:Line(iPart(Ans(X)
/ E6),iPart(E2fPart(
Ans(X)/E6)),iPart(E2
fPart(Ans(X)/E4)),E2
fPart(Ans(X)/E2),B
:End
```

```
PROGRAM: SPRITES3
:PrgmØFWINDOW
:(10102020,20203010
,30102015,20151010→L1
:For(I,1,10
:"UPDATE
:Ø→B:L1:PrgmØDRAWSPT
:L1+2(E4+1)→L1
:"DRAW
:I→B:L1:PrgmØDRAWSPT
:End
```

```
PROGRAM: SPRITES2
:PrgmØFWINDOW
:(10102020,20203010
,30102015,20151010→L1
:For(I,1,10
:"UPDATE
:L1+2(E4+1)→L1
:"DRAW
:L1:PrgmØDRAWSPT
:End
```

## Animation 2

- OPLOSSING: Gum in een lijn
- PROBLEEM: Overschrijft de achtergrond

```
PROGRAM: ØDRAWSPT
:For(X,1,dim(Ans
:Line(iPart(Ans(X)
/ E6),iPart(E2fPart(
Ans(X)/E6)),iPart(E2
fPart(Ans(X)/E4)),E2
fPart(Ans(X)/E2),B
:End
```

```
PROGRAM: SPRITES3
:PrgmØFWINDOW
:(10102020,20203010
,30102015,20151010→L1
:For(I,1,10
:"UPDATE
:Ø→B:L1:PrgmØDRAWSPT
:L1+2(E4+1)→L1
:"DRAW
:I→B:L1:PrgmØDRAWSPT
:End
```

```
PROGRAM: SPRITES2
:PrgmØFWINDOW
:(10102020,20203010
,30102015,20151010→L1
:For(I,1,10
:"UPDATE
:L1+2(E4+1)→L1
:"DRAW
:L1:PrgmØDRAWSPT
:End
```

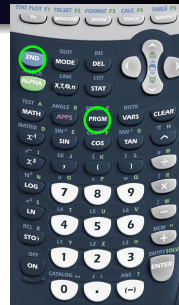
## Background image: `StorePic`

- Indien je iets stilstaands hebt, zoals een title screen of een background, dan wil je vaak dat het niet overschreven kan worden.
- Met `StorePic` kun je het opslaan in een plaatje (alleen mogelijk met het *hele* scherm).
- Met `RecallPic` kun je dat plaatje weer tekenen (bovenop wat er al op je scherm staat).
- Je kunt maximaal 10 `Pics` opslaan.



## Background image: StorePic

- Indien je iets stilstaands hebt, zoals een title screen of een background, dan wil je vaak dat het niet overschreven kan worden.
- Met **StorePic** kun je het opslaan in een plaatje (alleen mogelijk met het *hele* scherm).
- Met **RecallPic** kun je dat plaatje weer tekenen (bovenop wat er al op je scherm staat).
- Je kunt maximaal 10 **Pics** opslaan.

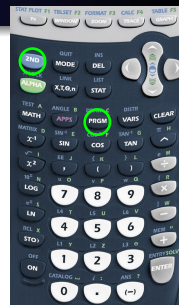


```
PROGRAM:SPRITEBG  
:PRGM0FWINDOW  
:Circle(10,10,30  
:StorePic 4
```

```
DRAW POINTS STO  
1:StorePic  
2:RecallPic  
3:StoreGDB(  
4:RecallGDB
```

## Background image: StorePic

- Indien je iets stilstaands hebt, zoals een title screen of een background, dan wil je vaak dat het niet overschreven kan worden.
- Met **StorePic** kun je het opslaan in een plaatje (alleen mogelijk met het *hele* scherm).
- Met **RecallPic** kun je dat plaatje weer tekenen (bovenop wat er al op je scherm staat).
- Je kunt maximaal 10 **Pics** opslaan.

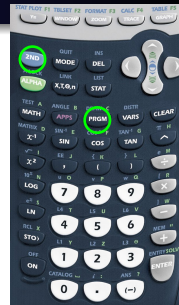


```
PROGRAM:SPRITEBG
:PRGM0FWINDOW
:Circle(10,10,30
:StorePic 4
:...do stuff...
:RecallPic 4
```

```
DRAW POINTS 510
1:StorePic
2:RecallPic
3:StoreGDB(
4:RecallGDB
```

## Background image: StorePic

- Indien je iets stilstaands hebt, zoals een title screen of een background, dan wil je vaak dat het niet overschreven kan worden.
- Met **StorePic** kun je het opslaan in een plaatje (alleen mogelijk met het *hele* scherm).
- Met **RecallPic** kun je dat plaatje weer tekenen (bovenop wat er al op je scherm staat).
- Je kunt maximaal 10 **Pics** opslaan.



```
PROGRAM:SPRITEBG
:PRGM0FWINDOW
:Circle(10,10,30
:StorePic 1
:...do stuff...
:RecallPic 1
```

```
DRAW POINTS 510
1:StorePic
2:RecallPic
3:StoreGDB(
4:RecallGDB
```

# Caching

- 'Caching' is het opslaan van een plaatje voor later gebruik.
- Stel dat je sprite over het scherm beweegt:
  - Indien je blijft tekenen zie je een heel spoor waar je sprite geweest is.
  - Zijn vorige positie moeten we clearen!
- Dit kan met 'caching':  
StorePic en RecallPic.

```
PROGRAM: SPRITES4
:Prgrm0FWINDOW
:Circle(10,10,30
:StorePic 4
: (10102020,20203010
,30102015,20151010→L1
:For(I,1,10
:"UPDATE
:L1+2(E4+1)→L1
:"DRAW

:L1:Prgrm0DRAWSPT
:End
```

# Caching

- 'Caching' is het opslaan van een plaatje voor later gebruik.
- Stel dat je sprite over het scherm beweegt:
  - Indien je blijft tekenen zie je een heel spoor waar je sprite geweest is.
  - Zijn vorige positie moeten we clearen!
- Dit kan met 'caching':  
StorePic en RecallPic.

```
PROGRAM: SPRITES4
:Prgrm0FWINDOW
:Circle(10,10,30
:StorePic 4
: (10102020,20203010
,30102015,20151010→L1
:For(I,1,10
:"UPDATE
:L1+2(E4+1)→L1
:"DRAW

:L1:Prgrm0DRAWSPT
:End
```

# Caching

- 'Caching' is het opslaan van een plaatje voor later gebruik.
- Stel dat je sprite over het scherm beweegt:
  - Indien je blijft tekenen zie je een heel spoor waar je sprite geweest is.
  - Zijn vorige positie moeten we clearen!
- Dit kan met 'caching':  
StorePic en RecallPic.

```
PROGRAM: SPRITES4
:Prgrm0FWINDOW
:Circle(10,10,30
:StorePic 4
: (10102020,20203010
,30102015,20151010→L1
:For(I,1,10
:"UPDATE
:L1+2(E4+1)→L1
:"DRAW

:L1:Prgrm0DRAWSPT
:End
```

# Caching

- 'Caching' is het opslaan van een plaatje voor later gebruik.
- Stel dat je sprite over het scherm beweegt:
  - Indien je blijft tekenen zie je een heel spoor waar je sprite geweest is.
  - Zijn vorige positie moeten we clearen!
- Dit kan met 'caching':  
StorePic en RecallPic.

```
PROGRAM: SPRITES4
:Prgrm0FWINDOW
:Circle(10,10,30
:StorePic 4
:(10102020,20203010
,30102015,20151010→L1
:For(I,1,10
:"UPDATE
:L1+2(E4+1)→L1
:"DRAW

:L1:Prgrm0DRAWSPT
:End
```

# Caching

- 'Caching' is het opslaan van een plaatje voor later gebruik.
- Stel dat je sprite over het scherm beweegt:
  - Indien je blijft tekenen zie je een heel spoor waar je sprite geweest is.
  - Zijn vorige positie moeten we clearen!
- Dit kan met 'caching':  
StorePic en RecallPic.

```
PROGRAM: SPRITES4
:Prgrm0FWINDOW
:Circle(10,10,30
:StorePic 4
:(10102020,20203010
,30102015,20151010→L1
:For(I,1,10
:"UPDATE
:L1+2(E4+1)→L1
:"DRAW
:ClrDraw:RecallPic 4
:L1:Prgrm0DRAWSPT
:End
```



## Animation 3: Clear small sprites

- Small sprites kunnen gecleared worden door spaties bovenop de sprite te tekenen.
- PROBLEEM: Werkt alleen makkelijk voor kleine sprites (max. 5 pixels hoog):  
Waardeloos voor dit voorbeeld!
- PROBLEEM: Achtergrond verdwijnt.

```
PROGRAM: SPRITES5
:prgmθFWINDOW
:Circle(10,10,30
:(10102020,20203010
,30102015,20151010→L1
:For(I,1,10
:"UPDATE
:L1+2(E4+1)→L1
:"DRAW
:Text(62-10,10,
"
: L1:prgmθDRAWSPT
:End
```

## Animation 3: Clear small sprites

- Small sprites kunnen gecleared worden door spaties bovenop de sprite te tekenen.
- PROBLEEM: Werkt alleen makkelijk voor kleine sprites (max. 5 pixels hoog):  
Waardeloos voor dit voorbeeld!
- PROBLEEM: Achtergrond verdwijnt.

```
PROGRAM: SPRITES5
:prgmθFWINDOW
:Circle(10,10,30
:(10102020,20203010
,30102015,20151010→L1
:For(I,1,10
:"UPDATE
:L1+2(E4+1)→L1
:"DRAW
:Text(62-10,10,
"
: L1:prgmθDRAWSPT
:End
```

## Animation 3: Clear small sprites

- Small sprites kunnen gecleared worden door spaties bovenop de sprite te tekenen.
- PROBLEEM: Werkt alleen makkelijk voor kleine sprites (max. 5 pixels hoog):  
Waardeloos voor dit voorbeeld!
- PROBLEEM: Achtergrond verdwijnt.

```
PROGRAM: SPRITES5
:prgmθFWINDOW
:Circle(10,10,30
:(10102020,20203010
,30102015,20151010→L1
:For(I,1,10
:"UPDATE
:L1+2(E4+1)→L1
:"DRAW
:Text(62-10,10,
"
: L1:prgmθDRAWSPT
:End
```

# Exercise

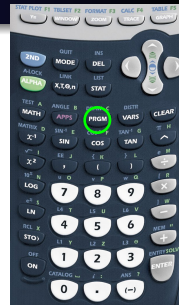
Maak zelf een simpele sprite die langzaam van links naar rechts beweegt.

# Outline

- 1 Graphics
  - Friendly window
  - Basic drawing functions
  - Sprites
- 2 Dynamic input
  - getKey
  - wait for input
  - Memory Leaks
- 3 Games!
  - Game Loop
  - Towards multiplayer...
- 4 Exercises
  - Exercises
  - Further Reading

# Check for key presses

- Met `getKey` kun je de *laatste* knop die ingedrukt is opvragen.
- Je krijgt een getal, welk naar een knop verwijst (zie figuur).



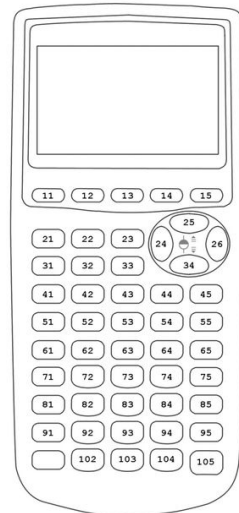
```

CTL █ EXEC
1:Input
2:Prompt
3:Disp
4:DispGraph
5:DispTable
6:Output(
7:getKey
  
```

# Check for key presses

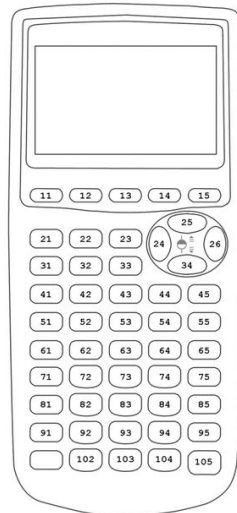
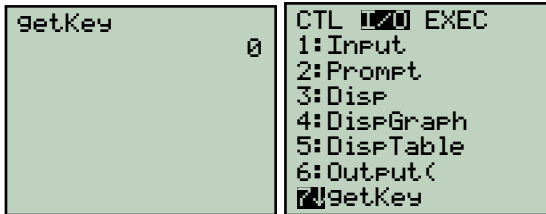
- Met `getKey` kun je de *laatste* knop die ingedrukt is opvragen.
- Je krijgt een getal, welk naar een knop verwijst (zie figuur).

```
CTL [Z] EXEC
1:Input
2:Prompt
3:Disp
4:DispGraph
5:DispTable
6:Output(
[Z]getKey
```



# Check for key presses



- Met `getKey` kun je de *laatste* knop die ingedrukt is opvragen.
- Je krijgt een getal, welk naar een knop verwijst (zie figuur).

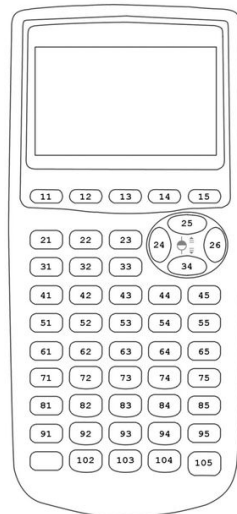




# Check for key presses



- Met `getKey` kun je de *laatste* knop die ingedrukt is opvragen.
- Je krijgt een getal, welk naar een knop verwijst (zie figuur).

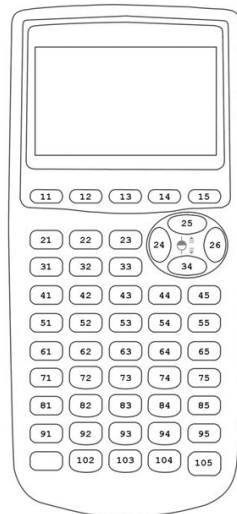
<code>getKey</code>	0	CTL  EXEC
<code>getKey</code>	0	1: Input
		2: Prompt
		3: Disp
		4: DispGraph
		5: DispTable
		6: Output(
		 getKey



# Check for key presses



- Met `getKey` kun je de *laatste* knop die ingedrukt is opvragen.
- Je krijgt een getal, welk naar een knop verwijst (zie figuur).

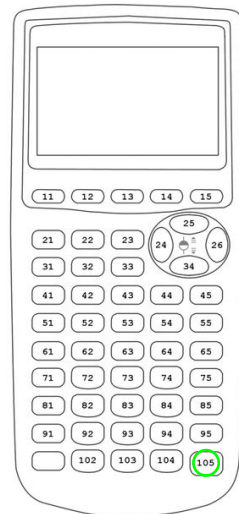
<code>getKey</code>	0	CTL  EXEC
<code>getKey</code>	0	1: Input
<code>getKey</code>	0	2: Prompt
<code>getKey</code>	0	3: Disp
		4: DispGraph
		5: DispTable
		6: Output(
		 getKey



# Check for key presses

- Met `getKey` kun je de *laatste* knop die ingedrukt is opvragen.
- Je krijgt een getal, welk naar een knop verwijst (zie figuur).

<code>getKey</code>		CTL  EXEC
<code>getKey</code>	0	1: Input
<code>getKey</code>	0	2: Prompt
<code>getKey</code>	0	3: Disp
<code>getKey</code>	0	4: DispGraph
<code>getKey</code>	0	5: DispTable
<code>getKey</code>	105	6: Output(
		 <code>getKey</code>



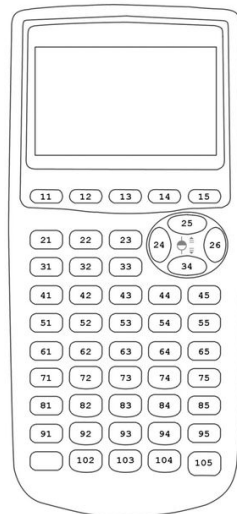
# Outline

- 1 Graphics
  - Friendly window
  - Basic drawing functions
  - Sprites
- 2 **Dynamic input**
  - getKey
  - **wait for input**
  - Memory Leaks
- 3 Games!
  - Game Loop
  - Towards multiplayer...
- 4 Exercises
  - Exercises
  - Further Reading

# Wait for keypress

- `PRgmGETKEY1` wacht tot een knop ingedrukt wordt.
- `PRgmGETKEY2` blijft lopen totdat er op `ENTER` gedrukt wordt.

```
PROGRAM: GETKEY1  
:Repeat Ans  
:getKey→X  
:Disp X  
:End
```

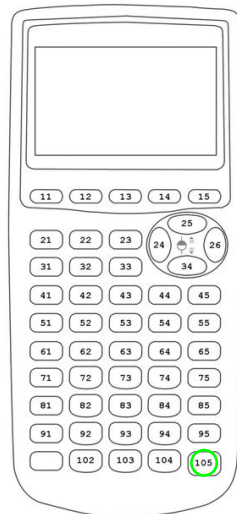


# Wait for keypress

- `PRgmGETKEY1` wacht tot een knop ingedrukt wordt.
- `PRgmGETKEY2` blijft lopen totdat er op `ENTER` gedrukt wordt.

```
PROGRAM:GETKEY2  
:0→K  
:While K≠105  
:getKey→K  
:If K≠0:Disp K  
:End
```

```
PROGRAM:GETKEY1  
:Repeat Ans  
:getKey→X  
:Disp X  
:End
```



# Hoe reageer je op specifieke input?

- Hoe kunnen we onze kennis van `getKey` gebruiken om te reageren op specifieke knoppen?
  - Bijv. reageer op pijltoetsen voor game/menu.

# Hoe reageer je op specifieke input?

- Hoe kunnen we onze kennis van `getKey` gebruiken om te reageren op specifieke knoppen?
  - Bijv. reageer op pijltoetsen voor game/menu.

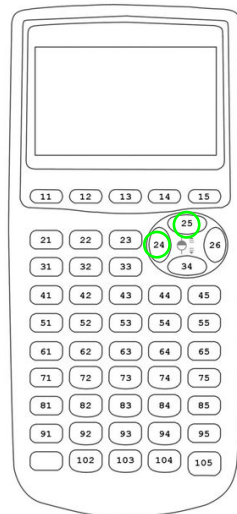
```
PROGRAM:AROWKEY1
:0→K:Repeat K
:getKey→X
:If K=25:Then
:Disp "MOVE UP
:End
:If K=24:Then
:Disp "MOVE <-
:End
:End
```



# Hoe reageer je op specifieke input?

- Hoe kunnen we onze kennis van `getKey` gebruiken om te reageren op specifieke knoppen?
  - Bijv. reageer op pijltoetsen voor game/menu.

```
PROGRAM:AROWKEY1
:0→K:Repeat K
:getKey→X
:If K=25:Then
:Disp "MOVE UP
:End
:If K=24:Then
:Disp "MOVE <-
:End
:End
```

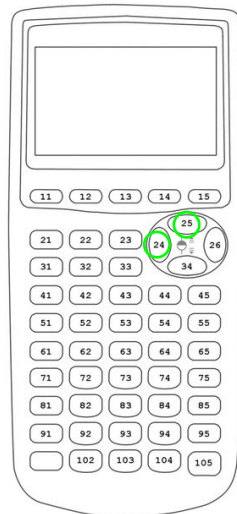


# Hoe reageer je op specifieke input?

- Hoe kunnen we onze kennis van `getKey` gebruiken om te reageren op specifieke knoppen?
  - Bijv. reageer op pijltoetsen voor game/menu.

```
PROGRAM:AROWKEY2
:Lbl 0:0→K
:Repeat K
:getKey→K
:End
:If K=25:Then
:Disp "MOVE UP
:Else
:If K=24:Then
:Disp "MOVE <-
:End:End
:Goto 0
```

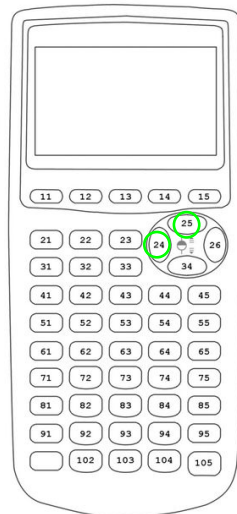
```
PROGRAM:AROWKEY1
:0→K:Repeat K
:getKey→X
:If K=25:Then
:Disp "MOVE UP
:End
:If K=24:Then
:Disp "MOVE <-
:End
:End
```



## Event → Action programming

- Een betere manier is PROGRAM:AROWKEY3.
- Merk op hoe raar de End's staan
  - Goto springt in het programma, dus klopt wel

```
PROGRAM:AROWKEY1
:0→K:Repeat K
:getKey→X
:If K=25:Then
:Disp "MOVE UP
:End
:If K=24:Then
:Disp "MOVE <-
:End
:End
```

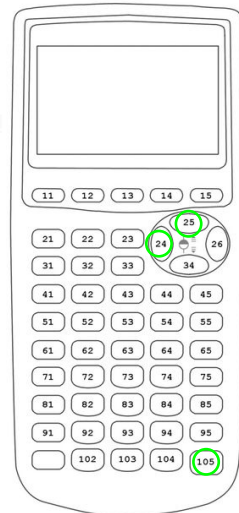


# Event → Action programming

- Een betere manier is PROGRAM:AROWKEY3.
- Merk op hoe raar de End's staan
  - Goto springt in het programma, dus klopt wel

```
PROGRAM:AROWKEY3
:If 1=1:Then
:Lb1 0:End
:getKey→K
:If K=0:Then
:Goto 0:End
:If K=25:Then
:Disp "MOVE UP
:Goto 0:End
:If K=24:Then
:Disp "MOVE <-
:Goto 0:End
:If K≠105:Then
:Goto 0
```

```
PROGRAM:AROWKEY1
:0→K:Repeat K
:getKey→X
:If K=25:Then
:Disp "MOVE UP
:End
:If K=24:Then
:Disp "MOVE <-
:End
:End
```

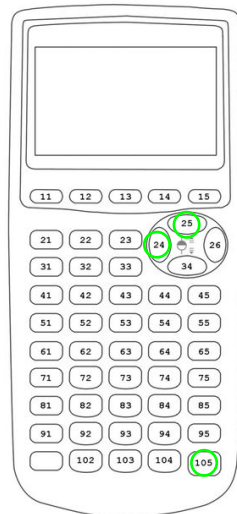


# Event → Action programming

- Een betere manier is PROGRAM:AROWKEY3.
- Merk op hoe raar de End's staan
  - Goto springt in het programma, dus klopt wel!

```
PROGRAM:AROWKEY3
:If 1=1:Then
:Lbl 0: [27]
:getKey→K
:If K=0:Then
:Goto 0:End
:If K=25:Then
:Disp "MOVE UP
:Goto 0:End
:If K=24:Then
:Disp "MOVE <-
:Goto 0:End
:If K≠105:Then
:Goto 0
```

```
PROGRAM:AROWKEY1
:0→K:Repeat K
:getKey→X
:If K=25:Then
:Disp "MOVE UP
:End
:If K=24:Then
:Disp "MOVE <-
:End
:End
```

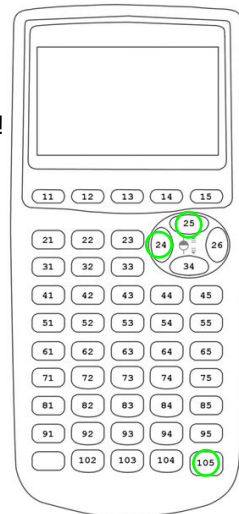


# Event → Action programming

- Een betere manier is PROGRAM:AROWKEY3.
- Merk op hoe raar de End's staan
  - Goto springt in het programma, dus klopt wel!

```
PROGRAM:AROWKEY3
:If 1=1:Then
:01 0:End
:getKey→K
:If K=0:Then
:Goto 0:End
:If K=25:Then
:Disp "MOVE UP
:Goto 0:End
:If K=24:Then
:Disp "MOVE <-
:Goto 0:End
:If K≠105:Then
:Goto 0
```

```
PROGRAM:AROWKEY1
:0→K:Repeat K
:getKey→X
:If K=25:Then
:Disp "MOVE UP
:End
:If K=24:Then
:Disp "MOVE <-
:End
:End
```



# Outline

- 1 Graphics
  - Friendly window
  - Basic drawing functions
  - Sprites
- 2 Dynamic input
  - getKey
  - wait for input
  - **Memory Leaks**
- 3 Games!
  - Game Loop
  - Towards multiplayer...
- 4 Exercises
  - Exercises
  - Further Reading

# Intermezzo: Memory Leaks

- Als je in een block statement (zoals `If` of `While`) gaat, onthoud de rekenmachine hoe diep je bent: dit heet de 'stack'.
- Wanneer je met `Goto` naar buiten springt, ga je steeds dieper en dieper.
- Dit kost memory...
- De memory leak verdwijnt pas zodra `Program` eindigt.
- <http://tibasicdev.wikidot.com/memory-leaks>

```
PROGRAM:AROWKEY3
:If 1=1:Then
:Lbl 0:END
:getKey→K
:If K=0:Then
:Goto 0:End
:If K=25:Then
:Disp "MOVE UP
:Goto 0:End
:If K=24:Then
:Disp "MOVE <-
:Goto 0:End
:If K≠105:Then
:Goto 0
```



# Intermezzo: Memory Leaks

- Als je in een block statement (zoals `If` of `While`) gaat, onthoud de rekenmachine hoe diep je bent: dit heet de 'stack'.
- Wanneer je met `Goto` naar buiten springt, ga je steeds dieper en dieper.
- Dit kost memory...
- De memory leak verdwijnt pas zodra `Program` eindigt.
- <http://tibasicdev.wikidot.com/memory-leaks>

```
PROGRAM:AROWKEY3
:If 1=1:Then
:Lbl 0:END
:getKey→K
:If K=0:Then
:Goto 0:End
:If K=25:Then
:Disp "MOVE UP
:Goto 0:End
:If K=24:Then
:Disp "MOVE ←-
:Goto 0:End
:If K≠105:Then
:Goto 0
```

# Intermezzo: Memory Leaks

- Als je in een block statement (zoals `If` of `While`) gaat, onthoud de rekenmachine hoe diep je bent: dit heet de 'stack'.
- Wanneer je met `Goto` naar buiten springt, ga je steeds dieper en dieper.
- Dit kost memory...
- De memory leak verdwijnt pas zodra `Program` eindigt.
- <http://tibasicdev.wikidot.com/memory-leaks>

```
:PROGRAM: MEMLEAK  
:Lbl A  
:While 1  
:Goto A  
:End
```

```
PROGRAM: AROWKEY3  
:If 1=1:Then  
:Lbl 0: END  
:getKey→K  
:If K=0:Then  
:Goto 0:End  
:If K=25:Then  
:Disp "MOVE UP  
:Goto 0:End  
:If K=24:Then  
:Disp "MOVE <-  
:Goto 0:End  
:If K≠105:Then  
:Goto 0
```

# Intermezzo: Memory Leaks

- Als je in een block statement (zoals `If` of `While`) gaat, onthoud de rekenmachine hoe diep je bent: dit heet de 'stack'.
- Wanneer je met `Goto` naar buiten springt, ga je steeds dieper en dieper.
- Dit kost memory...
- De memory leak verdwijnt pas zodra `Program` eindigt.
- <http://tibasicdev.wikidot.com/memory-leaks>

```
PRGM:MEMLEAK
```

```
:PROGRAM:MEMLEAK  
:Lb1 A  
:While 1  
:Goto A  
:End
```

```
PROGRAM:AROWKEY3  
:If 1=1:Then  
:Lb1 0:END  
:getKey→K  
:If K=0:Then  
:Goto 0:End  
:If K=25:Then  
:Disp "MOVE UP  
:Goto 0:End  
:If K=24:Then  
:Disp "MOVE <-  
:Goto 0:End  
:If K≠105:Then  
:Goto 0
```

# Intermezzo: Memory Leaks

- Als je in een block statement (zoals `If` of `While`) gaat, onthoud de rekenmachine hoe diep je bent: dit heet de 'stack'.
- Wanneer je met `Goto` naar buiten springt, ga je steeds dieper en dieper.
- Dit kost memory...Vertraagt je `Prgrm`...
- De memory leak verdwijnt pas zodra `Prgrm` eindigt.
- <http://tibasicdev.wikidot.com/memory-leaks>

```
PrgrmMEMLEAK
```

```
:PROGRAM:MEMLEAK  
:Lbl A  
:While 1  
:Goto A  
:End
```

```
PROGRAM:AROWKEY3  
:If 1=1:Then  
:Lbl 0: 270  
:getKey→K  
:If K=0:Then  
:Goto 0:End  
:If K=25:Then  
:Disp "MOVE UP  
:Goto 0:End  
:If K=24:Then  
:Disp "MOVE <-  
:Goto 0:End  
:If K≠105:Then  
:Goto 0
```

# Intermezzo: Memory Leaks

- Als je in een block statement (zoals `If` of `While`) gaat, onthoud de rekenmachine hoe diep je bent: dit heet de 'stack'.
- Wanneer je met `Goto` naar buiten springt, ga je steeds dieper en dieper.
- Dit kost memory...Vertraagt je `Prgrm`...  
En uiteindelijk crashed je `Prgrm`...
- De memory leak verdwijnt pas zodra `Prgrm` eindigt.
- <http://tibasicdev.wikidot.com/memory-leaks>

```
PrgrmMEMLEAK  
ERR:MEMORY
```

```
:PROGRAM: MEMLEAK  
:Lb1 A  
:While 1  
:Goto A  
:End
```

```
PROGRAM:AROWKEY3  
:If 1=1:Then  
:Lb1 0:END  
:getKey→K  
:If K=0:Then  
:Goto 0:End  
:If K=25:Then  
:Disp "MOVE UP  
:Goto 0:End  
:If K=24:Then  
:Disp "MOVE <-  
:Goto 0:End  
:If K≠105:Then  
:Goto 0
```

# Intermezzo: Memory Leaks

- Als je in een block statement (zoals `If` of `While`) gaat, onthoud de rekenmachine hoe diep je bent: dit heet de 'stack'.
- Wanneer je met `Goto` naar buiten springt, ga je steeds dieper en dieper.
- Dit kost memory...Vertraagt je `Prgrm`...  
En uiteindelijk crashed je `Prgrm`...
- De memory leak verdwijnt pas zodra `Prgrm` eindigt.
- <http://tibasicdev.wikidot.com/memory-leaks>

```
PrgrmMEMLEAK  
ERR:MEMORY
```

```
:PROGRAM: MEMLEAK  
:Lb1 A  
:While 1  
:Goto A  
:End
```

```
PROGRAM:AROWKEY3  
:If 1=1:Then  
:Lb1 0:END  
:getKey→K  
:If K=0:Then  
:Goto 0:End  
:If K=25:Then  
:Disp "MOVE UP  
:Goto 0:End  
:If K=24:Then  
:Disp "MOVE <-  
:Goto 0:End  
:If K≠105:Then  
:Goto 0
```

# Intermezzo: Memory Leaks

- Als je in een block statement (zoals `If` of `While`) gaat, onthoud de rekenmachine hoe diep je bent: dit heet de 'stack'.
- Wanneer je met `Goto` naar buiten springt, ga je steeds dieper en dieper.
- Dit kost memory...Vertraagt je `Prgrm`...  
En uiteindelijk crashed je `Prgrm`...
- De memory leak verdwijnt pas zodra `Prgrm` eindigt.
- <http://tibasicdev.wikidot.com/memory-leaks>

```
PrgrmMEMLEAK  
ERR:MEMORY
```

```
:PROGRAM:MEMLEAK  
:Lb1 A  
:While 1  
:Goto A  
:End
```

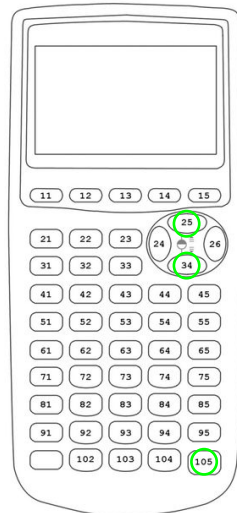
```
PROGRAM:AROWKEY3  
:If 1=1:Then  
:Lb1 0:END  
:getKey→K  
:If K=0:Then  
:Goto 0:End  
:If K=25:Then  
:Disp "MOVE UP  
:Goto 0:End  
:If K=24:Then  
:Disp "MOVE <-  
:Goto 0:End  
:If K≠105:Then  
:Goto 0
```

# Exercise

Maak een “custom menu”: een menu getekent met [DRAW]-functies bestuurt via `getKey`.

Maak het zo fancy als je wilt (cursor met animatie :)?), maar begin simpel.

```
WHAT PIE?  
- APPLE  
  BANANA CREAM  
  CHERRY
```



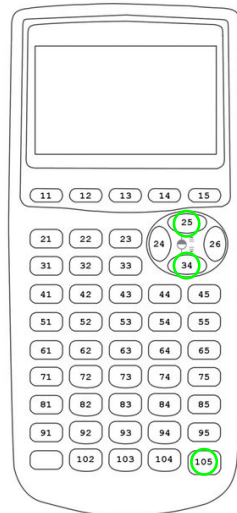


# Exercise

Maak een “custom menu”: een menu getekent met [DRAW]-functies bestuurt via `getKey`.

Maak het zo fancy als je wilt (cursor met animatie :)?), maar begin simpel.

```
WHAT PIE?  
APPLE  
- BANANA CREAM  
CHERRY
```

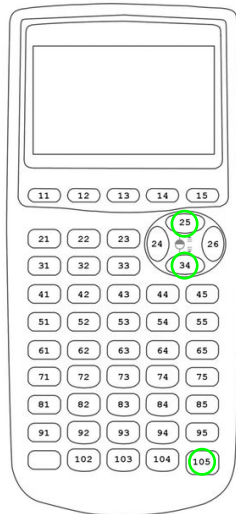


# Exercise

Maak een “custom menu”: een menu getekent met [DRAW]-functies bestuurt via `getKey`.

Maak het zo fancy als je wilt (cursor met animatie :)?), maar begin simpel.

```
WHAT PIE?  
APPLE  
BANANA CREAM  
- CHERRY
```



# Answer

```
PROGRAM: CUSTMENU
:ClrDraw:0→I
:Text(10,20,"APPLE
:Text(20,20,"BANANA CREAM
:Text(30,20,"CHERRY
:Text(40,20,"DERBY
:Text(50,20,"EMPANADA
:0→K:Repeat K=105
:getKey→K
:I→J
:If K=25:I-1→I
:If K=34:I+1→I
:min(4,max(0,I→I
:
:Text(10+10I,10,"-
:If I≠J:Text(10+10J,10,"_
:End
:Disp "YOU CHOSE",I+1
```

Met simpele animatie (8 frames):

```
: ...
:ClrDraw:0→I:0→A
: ...
:8fPart((A+1)/8→A
:Text(10+10I,10,"-
:Text(10+10I,6+A,"_
: ...
```

# Outline

- 1 Graphics
  - Friendly window
  - Basic drawing functions
  - Sprites
- 2 Dynamic input
  - getKey
  - wait for input
  - Memory Leaks
- 3 **Games!**
  - **Game Loop**
  - Towards multiplayer...
- 4 Exercises
  - Exercises
  - Further Reading

# Game Loop

- In tegenstelling tot veel van het voorgaande, wil je dat een game blijft spelen tot de gebruiker het afsluit.
  - Dit kun je bereiken met een 'infinite loop'.
  - Daarin moet een aantal dingen gebeuren:
- 
- Precieze vorm is afhankelijk van gametype:  
turn-based, real-time

# Game Loop

- In tegenstelling tot veel van het voorgaande, wil je dat een game blijft spelen tot de gebruiker het afsluit.
- Dit kun je bereiken met een 'infinite loop'.
- Daarin moet een aantal dingen gebeuren:
- Precieze vorm is afhankelijk van gametype:  
turn-based, real-time

```
PROGRAM: INFILOOP  
: While 1  
: do stuff  
: End
```

# Game Loop

- In tegenstelling tot veel van het voorgaande, wil je dat een game blijft spelen tot de gebruiker het afsluit.
  - Dit kun je bereiken met een 'infinite loop'.
  - Daarin moet een aantal dingen gebeuren:
- 
- Precieze vorm is afhankelijk van gametype:  
turn-based, real-time

```
PROGRAM: INFILOOP  
: While 1  
: do stuff  
: End
```

```
PROGRAM: GAMELOOP  
: "INIT GAME  
: While 1  
:  
  
:  
  
:  
  
: End
```

# Game Loop

- In tegenstelling tot veel van het voorgaande, wil je dat een game blijft spelen tot de gebruiker het afsluit.
- Dit kun je bereiken met een 'infinite loop'.
- Daarin moet een aantal dingen gebeuren:
  - ❶ Input (handle key presses)
- Precieze vorm is afhankelijk van gametype:  
turn-based, real-time

```
PROGRAM: INFILOOP
:While 1
:do stuff
:End
```

```
PROGRAM: GAMELOOP
:"INIT GAME
:While 1
:"INPUT
:getKey→K
:If K=...
:...
:

:
:End
```



# Game Loop

- In tegenstelling tot veel van het voorgaande, wil je dat een game blijft spelen tot de gebruiker het afsluit.
- Dit kun je bereiken met een 'infinite loop'.
- Daarin moet een aantal dingen gebeuren:
  - 1 Input (handle key presses)
  - 2 Update (rekenen)
- Precieze vorm is afhankelijk van gametype:  
turn-based, real-time

```
PROGRAM: INFILOOP
:While 1
:do stuff
:End
```

```
PROGRAM: GAMELOOP
:"INIT GAME
:While 1
:"INPUT
:getKey→K
:If K=...
:
:...
:"UPDATE
:L1+UT(E6+E2)+
VT(E4+1)→L1
:
:End
```

# Game Loop

- In tegenstelling tot veel van het voorgaande, wil je dat een game blijft spelen tot de gebruiker het afsluit.
- Dit kun je bereiken met een 'infinite loop'.
- Daarin moet een aantal dingen gebeuren:
  - 1 Input (handle key presses)
  - 2 Update (rekenen)
  - 3 Draw (tekenen)
- Precieze vorm is afhankelijk van gametype:  
turn-based, real-time

```
PROGRAM: INFILOOP
:While 1
:do stuff
:End
```

```
PROGRAM: GAMELOOP
:"INIT GAME
:While 1
:"INPUT
:getKey→K
:If K=...
:...
:"UPDATE
:L1+UT(E6+E2)+
VT(E4+1)→L1
:"DRAW
:L1:prgmDRAWSP
:End
```

# Game Loop

- In tegenstelling tot veel van het voorgaande, wil je dat een game blijft spelen tot de gebruiker het afsluit.
- Dit kun je bereiken met een 'infinite loop'.
- Daarin moet een aantal dingen gebeuren:
  - 1 Input (handle key presses)
  - 2 Update (rekenen)
  - 3 Draw (tekenen)
- Precieze vorm is afhankelijk van gametype:  
turn-based, real-time

```
PROGRAM: INFILOOP
:While 1
:do stuff
:End
```

```
PROGRAM: GAMELOOP
:"INIT GAME
:While 1
:"INPUT
:getKey→K
:If K=...
:...
:"UPDATE
:L1+UT(E6+E2)+
VT(E4+1)→L1
:"DRAW
:L1:prgmDRAWSP
:End
```

# Real-Time Games

- In een real-time game gaat de game door, of je nou op een knop drukt of niet.

```
PROGRAM: REALTIME
: "INIT GAME
: While 1
: "INPUT
: getKey→K
: If K=0: Then
: Goto UD: End
: If K=25: Then
: V+1→V
: Goto UD: End
: If K=34: Then
: V-1→V
: Goto UD: End
: If K=24 or
K=26: Then
: U-25+K→U
: Goto UD: End
```

```
: If K=105: Then
: Goto X
: "UPDATE
: Lb1 UD: End
: L1+U(E6+E2)+
V(E4+1)→L1
: "DRAW
: L1: prgmθDRAWSPT
: End
: Lb1 X
: Clean stuff up
here, reset graph
axis etc.
```

# Real-Time Games

- In een real-time game gaat de game door, of je nou op een knop drukt of niet.

```
PROGRAM: REALTIME
: "INIT GAME
: While 1
: "INPUT
: getKey→K
: If K=0: Then
: Goto UD: End
: If K=25: Then
: V+1→V
: Goto UD: End
: If K=34: Then
: V-1→V
: Goto UD: End
: If K=24 or
K=26: Then
: U-25+K→U
: Goto UD: End
```

```
: If K=105: Then
: Goto X
: "UPDATE
: Lb1 UD: End
: L1+U(E6+E2)+
V(E4+1)→L1
: "DRAW
: L1: prgmθDRAWSPT
: End
: Lb1 X
: Clean stuff up
here, reset graph
axis etc.
```

# Turn-Based Games

- In een turn-based game gebeurt er alleen iets nieuws wanneer er op de knop wordt gedrukt.

```
PROGRAM: TURNGAME
: "INIT GAME
: While 1
: "INPUT
: Repeat. Ans
: getKey→K
: and
: If K=25: Then
: 1→V
: Goto UD: End
: If K=34: Then
: -1→V
: Goto UD: End
: If K=24 or
K=26: Then
: K-25→U
: Goto UD: End
```

```
: If K=105: Then
: Goto X
: "UPDATE
: Lbl UD: End
: L1+U(E6+E2)+
V(E4+1)→L1
: 0→U: 0→V
: "DRAW
: L1: prgm0DRAWSPT
: End
: Lbl X
: Clean stuff up
here, reset graph
axis etc.
```

# Turn-Based Games

- In een turn-based game gebeurt er alleen iets nieuws wanneer er op de knop wordt gedrukt.

```
PROGRAM: TURNGAME
: "INIT GAME
: While 1
: "INPUT
: Repeat. Ans
: getKey→K
: and
: If K=25: Then
: 1→V
: Goto UD: End
: If K=34: Then
: -1→V
: Goto UD: End
: If K=24 or
K=26: Then
: K-25→U
: Goto UD: End
```

```
: If K=105: Then
: Goto X
: "UPDATE
: Lb1 UD: End
: L1+U(E6+E2)+
V(E4+1)→L1
: 0→U: 0→V
: "DRAW
: L1: prgm0DRAWSPT
: End
: Lb1 X
: Clean stuff up
here, reset graph
axis etc.
```

# Turn-Based Games with Animations

- Indien animaties door moeten gaan, moet je iets slimmer zijn...

```
PROGRAM: TURNGAM2
: "INIT GAME
: While 1
: "INPUT
: getKey→K
: If K=0: Then
: If K=25: Then
: 1→V
: Goto UD: End
: If K=34: Then
: -1→V
: Goto UD: End
: If K=24 or
K=26: Then
: K-25→U
: Goto UD: End
```

```
: If K=105: Then
: Goto X
: Lbl UD: End
: "PROCESS TURN
: L1+U(E6+E2)+
V(E4+1)→L1
: 0→U: 0→V
: End
: "ANIMATE
: Update sprite
animation here
: "DRAW
: L1: prgm0DRAWSPT
: End
: Lbl X
: Clean stuff up
here, reset graph
axis etc.
```



# Turn-Based Games: Wait for Animation

- Het kan logisch zijn om te wachten tot een animatie klaar is in een turn-based game, voordat de volgende zet gedaan mag worden.
- Dit voorbeeld negeert de input (K) totdat de animatie (A) klaar is.

```
PROGRAM: TURNGAM3
: "INIT GAME
: While 1
: "INPUT
: getKey→K
:
: If K≠0: Then
:
: If K=25: Then
: 1→V
: Goto UD: End
: If K=34: Then
: -1→V
: Goto UD: End
: If K=24 or
K=26: Then
: K-25→U
: Goto UD: End
```

```
: If K=105: Then
: Goto X
: Lbl UD: End
: "PROCESS TURN
: L1+U(E6+E2)+
V(E4+1)→L1
: 0→U: 0→V
: End
: "ANIMATE
: Update sprite
animation here
:
: "DRAW
: L1: prgm0DRAWSPT
: End
: Lbl X
: Clean stuff up
here, reset graph
axis etc.
```

# Turn-Based Games: Wait for Animation

- Het kan logisch zijn om te wachten tot een animatie klaar is in een turn-based game, voordat de volgende zet gedaan mag worden.
- Dit voorbeeld negeert de input (K) totdat de animatie (A) klaar is.

```
PROGRAM: TURNGAM3
: "INIT GAME
: While 1
: "INPUT
: getKey→K
: If A=1:K=0
: If K≠0:Then
: 1→A
: If K=25:Then
: 1→V
: Goto UD:End
: If K=34:Then
: -1→V
: Goto UD:End
: If K=24 or
K=26:Then
: K-25→U
: Goto UD:End
```

```
: If K=105:Then
: Goto X
: Lbl UD:End
: "PROCESS TURN
: L1+U(E6+E2)+
V(E4+1)→L1
: 0→U: 0→V
: End
: "ANIMATE
: Update sprite
animation here
: If animation
finished: 0→A
: "DRAW
: L1: prgm0DRAWSPT
: End
: Lbl X
: Clean stuff up
here, reset graph
axis etc.
```

# Outline

- 1 Graphics
  - Friendly window
  - Basic drawing functions
  - Sprites
- 2 Dynamic input
  - getKey
  - wait for input
  - Memory Leaks
- 3 **Games!**
  - Game Loop
  - **Towards multiplayer...**
- 4 Exercises
  - Exercises
  - Further Reading

# Wat is een multiplayer game?

- Meerdere spelers spelen hetzelfde spel:
  - Tegen elkaar (PvP)
  - Samen met elkaar (coop)
- Meerdere manieren:
  - One Device
  - Two Devices
- Ik geef slechtst een korte overview. Details staan hier:
  - <http://tibasicdev.wikidot.com/multiplayer>

# Wat is een multiplayer game?

- Meerdere spelers spelen hetzelfde spel:
  - Tegen elkaar (PvP)
  - Samen met elkaar (coop)
- Meerdere manieren:
  - One Device
  - Two Devices
- Ik geef slechtst een korte overview. Details staan hier:
  - <http://tibasicdev.wikidot.com/multiplayer>

# Wat is een multiplayer game?

- Meerdere spelers spelen hetzelfde spel:
  - Tegen elkaar (PvP)
  - Samen met elkaar (coop)
- Meerdere manieren:
  - One Device
  - Two Devices
- Ik geef slechtst een korte overview. Details staan hier:
  - <http://tibasicdev.wikidot.com/multiplayer>

# Wat is een multiplayer game?

- Meerdere spelers spelen hetzelfde spel:
  - Tegen elkaar (PvP)
  - Samen met elkaar (coop)
- Meerdere manieren:
  - One Device
  - Two Devices
- Ik geef slechtst een korte overview. Details staan hier:
  - <http://tibasicdev.wikidot.com/multiplayer>

# Wat is een multiplayer game?

- Meerdere spelers spelen hetzelfde spel:
  - Tegen elkaar (PvP)
  - Samen met elkaar (coop)
- Meerdere manieren:
  - One Device
  - Two Devices
- Ik geef slechtst een korte overview. Details staan hier:
  - <http://tibasicdev.wikidot.com/multiplayer>



# Multiplayer: One Device

- Dit is vrijwel hetzelfde als een single player game, maar nu:
  - Andere knoppen besturen andere sprites: meer `If K=...` statements.
- BELANGRIJK: Slechts 1 knop kan tegelijk ingedrukt worden.
  - PROBLEEM: Indien je `Prgm` langzaam is, dan komen niet beide spelers aan de beurt; alleen degene die het snelste drukt.
  - PAS OP: Pijltoetsen kunnen ingedrukt worden gehouden. Gebruik die knoppen dus NIET.
  - Geen probleem voor turn-based games!  
Turn-based is een aanrader voor multiplayer!

# Multiplayer: One Device

- Dit is vrijwel hetzelfde als een single player game, maar nu:
  - Andere knoppen besturen andere sprites: meer `If K=...` statements.
- BELANGRIJK: Slechts 1 knop kan tegelijk ingedrukt worden.
  - PROBLEEM: Indien je `Prgm` langzaam is, dan komen niet beide spelers aan de beurt; alleen degene die het snelste drukt.
  - PAS OP: Pijltoetsen kunnen ingedrukt worden gehouden. Gebruik die knoppen dus NIET.
  - Geen probleem voor turn-based games!  
Turn-based is een aanrader voor multiplayer!

# Multiplayer: One Device

- Dit is vrijwel hetzelfde als een single player game, maar nu:
  - Andere knoppen besturen andere sprites: meer `If K=...` statements.
- **BELANGRIJK:** Slechts 1 knop kan tegelijk ingedrukt worden.
  - PROBLEEM: Indien je `FROM` langzaam is, dan komen niet beide spelers aan de beurt; alleen degene die het snelste drukt.
  - PAS OP: Pijltoetsen kunnen ingedrukt worden gehouden. Gebruik die knoppen dus NIET.
  - Geen probleem voor turn-based games!  
Turn-based is een aanrader voor multiplayer!

# Multiplayer: One Device

- Dit is vrijwel hetzelfde als een single player game, maar nu:
  - Andere knoppen besturen andere sprites: meer `If K=...` statements.
- BELANGRIJK: Slechts 1 knop kan tegelijk ingedrukt worden.
  - PROBLEEM: Indien je **FROM** langzaam is, dan komen niet beide spelers aan de beurt; alleen degene die het snelste drukt.
  - PAS OP: Pijltoetsen kunnen ingedrukt worden gehouden. Gebruik die knoppen dus NIET.
  - Geen probleem voor turn-based games!  
Turn-based is een aanrader voor multiplayer!

# Multiplayer: One Device

- Dit is vrijwel hetzelfde als een single player game, maar nu:
  - Andere knoppen besturen andere sprites: meer `If K=...` statements.
- BELANGRIJK: Slechts 1 knop kan tegelijk ingedrukt worden.
  - PROBLEEM: Indien je **FROM** langzaam is, dan komen niet beide spelers aan de beurt; alleen degene die het snelste drukt.
  - PAS OP: Pijltoetsen kunnen ingedrukt worden gehouden. Gebruik die knoppen dus NIET.
  - Geen probleem voor turn-based games!  
Turn-based is een aanrader voor multiplayer!

# Multiplayer: One Device

- Dit is vrijwel hetzelfde als een single player game, maar nu:
  - Andere knoppen besturen andere sprites: meer `If K=...` statements.
- BELANGRIJK: Slechts 1 knop kan tegelijk ingedrukt worden.
  - PROBLEEM: Indien je ~~FROM~~ langzaam is, dan komen niet beide spelers aan de beurt; alleen degene die het snelste drukt.
  - PAS OP: Pijltoetsen kunnen ingedrukt worden gehouden. Gebruik die knoppen dus NIET.
  - Geen probleem voor turn-based games!  
Turn-based is een aanrader voor multiplayer!

# Multiplayer: Two Devices

- Het principe is simpel:
  - Connect een linkabel.
  - Beide spelers starten hetzelfde `prog`.
  - Beide spelers kunnen hun eigen knoppen gebruiken.
- Linken van variabelen gaat via de `GetCalc(varname)` functie.
- PROBLEEM: Linken kan alleen wanneer één rekenmachine niets doet, bijvoorbeeld door `Pause`.
- Daarom alleen mogelijk voor turn-based games.
  - Maar dat kan net zo goed op één device...

# Multiplayer: Two Devices

- Het principe is simpel:
  - Connect een linkabel.
  - Beide spelers starten hetzelfde `prog`.
  - Beide spelers kunnen hun eigen knoppen gebruiken.
- Linken van variabelen gaat via de `GetCalc(varname)` functie.
- PROBLEEM: Linken kan alleen wanneer één rekenmachine niets doet, bijvoorbeeld door `Pause`.
- Daarom alleen mogelijk voor turn-based games.
  - Maar dat kan net zo goed op één device...



# Multiplayer: Two Devices

- Het principe is simpel:
  - Connect een linkabel.
  - Beide spelers starten hetzelfde **Program**.
  - Beide spelers kunnen hun eigen knoppen gebruiken.
- Linken van variabelen gaat via de `GetCalc(varname)` functie.
- PROBLEEM: Linken kan alleen wanneer één rekenmachine niets doet, bijvoorbeeld door **Pause**.
- Daarom alleen mogelijk voor turn-based games.
  - Maar dat kan net zo goed op één device...

# Multiplayer: Two Devices

- Het principe is simpel:
  - Connect een linkabel.
  - Beide spelers starten hetzelfde **Program**.
  - Beide spelers kunnen hun eigen knoppen gebruiken.
- Linken van variabelen gaat via de `GetCalc(varname)` functie.
- PROBLEEM: Linken kan alleen wanneer één rekenmachine niets doet, bijvoorbeeld door **Pause**.
- Daarom alleen mogelijk voor turn-based games.
  - Maar dat kan net zo goed op één device...

# Multiplayer: Two Devices

- Het principe is simpel:
  - Connect een linkabel.
  - Beide spelers starten hetzelfde **Program**.
  - Beide spelers kunnen hun eigen knoppen gebruiken.
- Linken van variabelen gaat via de `GetCalc(varname)` functie.
- PROBLEEM: Linken kan alleen wanneer één rekenmachine niets doet, bijvoorbeeld door **Pause**.
- Daarom alleen mogelijk voor turn-based games.
  - Maar dat kan net zo goed op één device...

# Multiplayer: Two Devices

- Het principe is simpel:
  - Connect een linkabel.
  - Beide spelers starten hetzelfde **Program**.
  - Beide spelers kunnen hun eigen knoppen gebruiken.
- Linken van variabelen gaat via de `GetCalc(varname)` functie.
- PROBLEEM: Linken kan alleen wanneer één rekenmachine niets doet, bijvoorbeeld door **Pause**.
- Daarom alleen mogelijk voor turn-based games.
  - Maar dat kan net zo goed op één device...

# Multiplayer: Two Devices

- Het principe is simpel:
  - Connect een linkabel.
  - Beide spelers starten hetzelfde **Program**.
  - Beide spelers kunnen hun eigen knoppen gebruiken.
- Linken van variabelen gaat via de `GetCalc(varname)` functie.
- PROBLEEM: Linken kan alleen wanneer één rekenmachine niets doet, bijvoorbeeld door **Pause**.
- Daarom alleen mogelijk voor turn-based games.
  - Maar dat kan net zo goed op één device...

# Multiplayer: Two Devices

- Het principe is simpel:
  - Connect een linkabel.
  - Beide spelers starten hetzelfde **Program**.
  - Beide spelers kunnen hun eigen knoppen gebruiken.
- Linken van variabelen gaat via de `GetCalc(varname)` functie.
- PROBLEEM: Linken kan alleen wanneer één rekenmachine niets doet, bijvoorbeeld door **Pause**.
- Daarom alleen mogelijk voor turn-based games.
  - Maar dat kan net zo goed op één device...

# Outline

- 1 Graphics
  - Friendly window
  - Basic drawing functions
  - Sprites
- 2 Dynamic input
  - getKey
  - wait for input
  - Memory Leaks
- 3 Games!
  - Game Loop
  - Towards multiplayer...
- 4 Exercises
  - Exercises
  - Further Reading

# Exercises...?

Nu heb je geen exercises meer nodig, toch?

- ① Bedenk een idee voor een spel.
- ② Versimpel je idee A LOT.
  - Je idee was geweldig!, maar als je het niet eerst versimpelt gaat het je nooit lukken.
  - One step at a time! Begin altijd simpel.
- ③ Maak pseudocode.
- ④ Maak de game loop.
- ⑤ Maak een simpele sprite om mee te beginnen.
- ⑥ Implementeer een minimaal-werkende game.
- ⑦ Breid de game nu langzaam uit met meer features.



# Outline

- 1 Graphics
  - Friendly window
  - Basic drawing functions
  - Sprites
- 2 Dynamic input
  - getKey
  - wait for input
  - Memory Leaks
- 3 Games!
  - Game Loop
  - Towards multiplayer...
- 4 Exercises
  - Exercises
  - Further Reading

## Further Reading

- Op deze website staat veel informatie over alles wat je maar wilt weten:
  - <http://tibasicdev.wikidot.com/home>
  - <http://tibasicdev.wikidot.com/graphics>
  - <http://tibasicdev.wikidot.com/maps>
  - <http://tibasicdev.wikidot.com/cryptography>
  - <http://tibasicdev.wikidot.com/multiplayer>
- Of stel je vraag op het officiële forum:
  - <http://tibasicdev.wikidot.com/forum:home>