
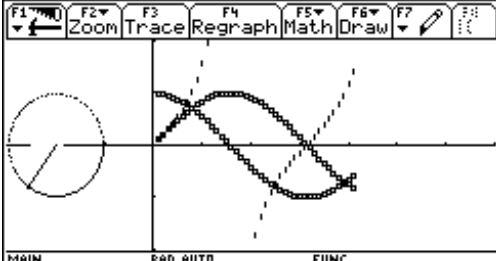


Unità di lavoro: Funzioni goniometriche con Ti92

Il programma “angoli” ha lo scopo di visualizzarne la collocazione degli angoli sul cerchio goniometrico e quello di calcolare e visualizzare le relative funzioni goniometriche.

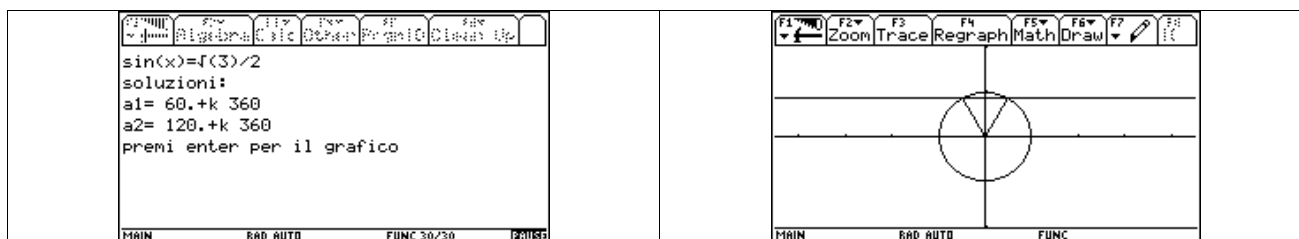
<p>Nome del programma al quale viene passato il parametro gr corrispondente alla misura in gradi</p> <p>Pulisce lo schermo</p> <p>Pulisce lo schermo grafico</p> <p>Riporta sullo schermo l'angolo</p> <p>Converte in radianti</p> <p>Scrive l'angolo in radianti</p> <p>Calcola e scrive il valore del seno</p> <p>“ “ “ coseno</p> <p>“ “ “ della tangente</p> <p>segnala di premere Enter: l'esecuzione del programma viene sospesa prima di passare allo schermo grafico</p> <p>Vengono definite le coordinate minime e massime di schermo</p> <p>Traccia un cerchio di centro $-2; 0$ e raggio 1</p> <p>Inizia il ciclo di animazione col parametro i che varia da 0.1 a rd con passo di 0.1</p> <p>assegna alle variabili x e y i valori del $\cos(i)$ e del $\sin(i)$. i rappresenta l'angolo variabile</p> <p>Traccia il raggio che varia sul cerchio fino a formare l'angolo rd</p> <p>Traccia un circoletto di raggio 0.05 per la funzione \cos</p> <p>Idem per la funzione \sin</p> <p>Idem per la funzione \tan</p> <p>Questo ciclo di j da 1 a 20 ha il solo scopo di far tardare la cancellazione del raggio variabile</p> <p>Cancella il raggio variabile</p> <p>Termina il ciclo (quando si raggiunge il valore rd)</p> <p>Traccia il raggio relativo alla posizione finale</p> <p>Fine programma.</p> <p>Digitando in HOME : <code>angoli(240)</code> si ottiene</p>	<pre>angoli(gr) Prgm ClrIO ClrDraw Disp "funzioni goniometriche di "&string(gr) gr/180 * π → rd Disp "angolo in radianti "&string(rd) Disp "seno = "&string(sin(rd)) Disp "coseno = "&string(cos(rd)) Disp "tangente = "&string(tan(rd)) Disp "premi enter per il grafico" Pause -3 → xmin 7 → xmax -2 → ymin 2 → ymax Circle -2,0,1 For i,.1,rd,.1 cos(i) → x sin(i) → y Line -2,0,x-2,y,1 Circle i,x,.05 Circle i,y,.05 Circle i,tan(i),.04 For j,1,20,1 EndFor Line -2,0,x-2,y,0 EndFor Line 2,0,x-2,y,1 EndPrgm</pre>
	

Il seguente programma serve per risolvere le equazioni elementari in $\sin(x)$.
 Il bordo attorno ad alcune righe verrà spiegato più avanti.

Programma eqsin(a)

Nome del programma al quale viene passato il parametro a corrispondente alla misura in gradi	eqsin(a)
Pulisce lo schermo	Prgm
Pulisce lo schermo grafico	ClrO
Riporta sullo schermo l'equazione	ClrDraw
Controlla che il valore a sia compreso fra -1 e 1	Disp "sin(x)="+string(a)
Calcola il valore dell'angolo in radianti	If $-1 \leq a$ and $1 \geq a$ Then
Calcola l'angolo supplementare	Disp "soluzioni: "
Converte in gradi il primo angolo	approx(sin ⁻¹ (a) → x1
Calcola il valore del supplementare in gradi	$\pi - x1 \rightarrow x2$
Scriva le soluzioni	$x1 * 180 / \pi \rightarrow gr1$
Segnala di premere Enter: l'esecuzione del programma viene sospesa prima di passare allo schermo grafico	180-gr1 → gr2
Vengono definite le coordinate minime e massime di schermo	Disp "a1= " + string(gr1) + "k 360"
Traccia un cerchio di centro (0;0) e raggio 1	Disp "a2= " + string(gr2) + "k 360"
Traccia una linea ad altezza = a	Disp "premi enter per il grafico"
Traccia i due segmenti che uniscono il centro delle coordinate ai due punti che rappresentano le soluzioni.	Pause
Scriva "impossibile" se $-1 \leq a \leq 1$	- 4.5 → xmin
	4.5 → xmax
	-2 → ymin
	2 → ymax
	Circle 0,0,1
	Line xmin,a,xmax,a
	Line 0,0,cos(x1),sin(x1)
	Line 0,0,cos(x2),sin(x2)
	Else
	Disp "impossibile"
	EndIf
	EndPrgm

Digitando in HOME eqsin($\sqrt{3}/2$) si ottiene :



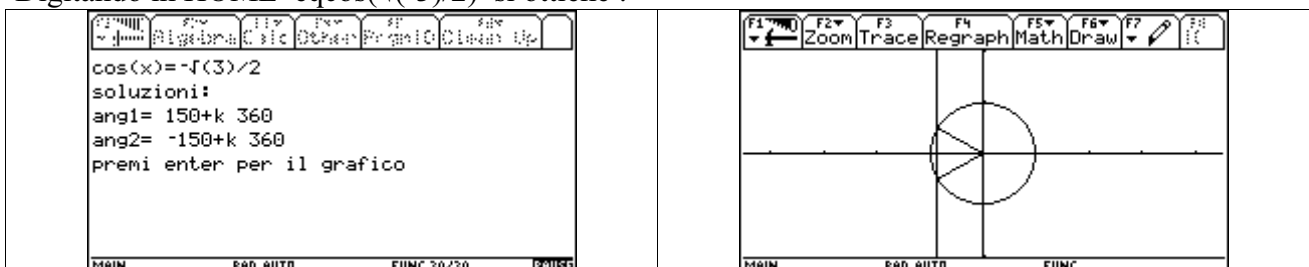
Esercizio: salvare il programma (F1 - Save Copy as) col nome **eqcos** e modificare opportunamente le istruzioni incorniciate del programma in modo che risolva le equazioni elementari in coseno.

Soluzione:

Il programma **eqcos** è del tutto analogo a eqsin, occorre cambiare solo le linee con bordo conviene quindi salvare il programma eqcos usando il comando **F1 - Save Copy as e**, alla richiesta, assegnare il nome eqcos. Le righe da cambiare sono quelle nei riquadri.

Nome del programma al quale viene passato il parametro a corrispondente alla misura in gradi	eqcos(a)
Pulisce lo schermo	Prgm
Pulisce lo schermo grafico	ClrO
Riporta sullo schermo l'equazione	ClrDraw
	Disp "cos(x)="+string(a)
Controlla che il valore a sia compreso fra -1 e 1	If $-1 \leq a$ and $1 \geq a$ Then
	Disp "soluzioni: "
Calcola il valore dell'angolo in radianti	approx(cos ⁻¹ (a)) → x1
Calcola l'angolo opposto	- x1 → x2
Converte in gradi il primo angolo	x1*180/π → gr1
Calcola il valore dell'opposto in gradi	-gr1 → gr2
Scrivere le soluzioni	Disp "a1= "&string(gr1)&"k 360"
	Disp "a2= "&string(gr2)&"k 360"
	Disp "premi enter per il grafico"
Segnala di premere Enter: l'esecuzione del programma viene sospesa prima di passare allo schermo grafico	Pause
Vengono definite le coordinate minime e massime di schermo	- 4.5 → xmin
	4.5 → xmax
	-2 → ymin
	2 → ymax
	Circle 0,0,1
Traccia un cerchio di centro (0;0) e raggio 1	Line a, ymin,a,ymax
Traccia una linea verticale a distanza dal centro = a	Line 0,0,cos(x1),sin(x1)
	Line 0,0,cos(x2),sin(x2)
Traccia i due segmenti che uniscono il centro delle coordinate ai due punti che rappresentano le soluzioni.	Else
	Disp "impossibile"
	EndIf
Scrivere "impossibile" se $-1 \leq a \leq 1$	EndPrgm

Digitando in HOME eqcos($\sqrt{(-3)/2}$) si ottiene :



Per ottenere i programmi che risolvono le disequazioni conviene creare le copie dei tre programmi *eqsin*, *eqcos* ed *eqtan* (F1- Save Copy As) assegnando i nomi *dissin*, *discos* e *distan* , quindi basta inserire le istruzioni incorniciate.

Programma *dissin(a)*

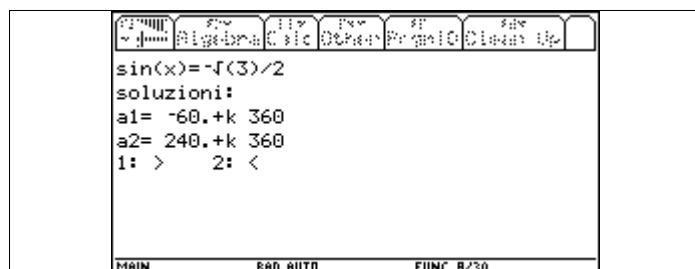
Nome del programma al quale viene passato il parametro a corrispondente alla misura in gradi
 Pulisce lo schermo
 Pulisce lo schermo grafico
 Riporta sullo schermo l'equazione
 Controlla che il valore a sia compreso fra -1 e 1

Calcola il valore dell'angolo in radianti
 Calcola l'angolo supplementare
 Converte in gradi il primo angolo
 Calcola il valore del supplementare in gradi
 Scrive le soluzioni

Chiede la scelta fra $>$ e $<$ ossia $\sin(x) > a$ o $\sin(x) < a$

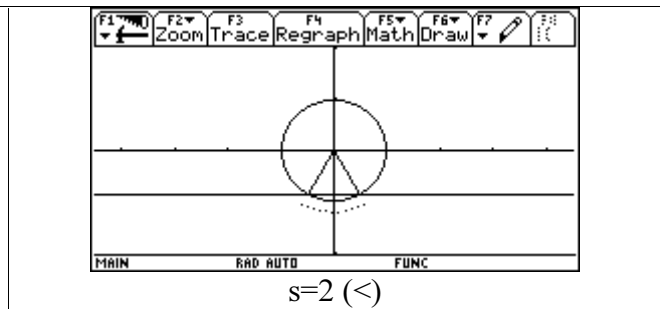
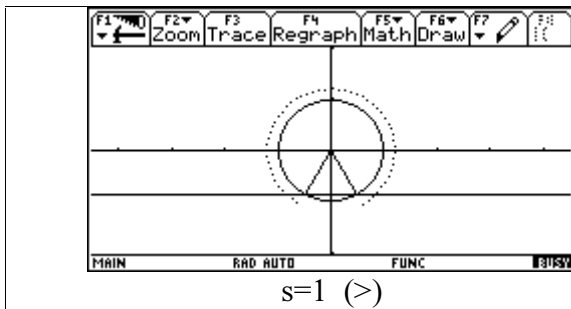
Traccia un cerchio di centro $(0;0)$ e raggio 1
 Traccia una linea ad altezza $= a$
 Traccia i due segmenti che uniscono il centro delle coordinate ai due punti che rappresentano le soluzioni.
 Blocco che traccia le soluzioni della disequazione

Dissin $(-\sqrt{3}/2)$



```

dissin(a)
Prgm
ClrIO
ClrDraw
Disp "sin(x)="&string(a)
If -1 ≤ a and 1 ≥ a Then
Disp "soluzioni: "
approx(sin-1(a)) → x1
π - x1 → x2
x1*180/π → gr1
180-gr1 → gr2
Disp "a1= "&string(gr1)&" +k 360"
Disp "a2= "&string(gr2)&" +k 360"
input "1: > 2: < ",s
-4.5 → xmin
4.5 → xmax
-2 → ymin
2 → ymax
Circle 0,0,1
Line xmin,a,xmax,a
Line 0,0,cos(x1),sin(x1)
Line 0,0,cos(x2),sin(x2)
If s=1 then
x1 → h1
x2 → h2
else
x2 → h1
2*π+x1 → h2
endif
For i,h1,h2,0.1
PtOn 1.2*cos(i),1.2*sin(i)
EndFor
Else
Disp "impossibile"
EndIf
EndPrgm
  
```



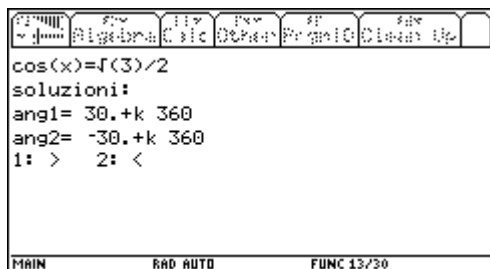
Programma **discos(a)**

Nome del programma al quale viene passato il parametro a corrispondente alla misura in gradi
 Pulisce lo schermo
 Pulisce lo schermo grafico
 Riporta sullo schermo l'equazione
 Controlla che il valore a sia compreso fra -1 e 1

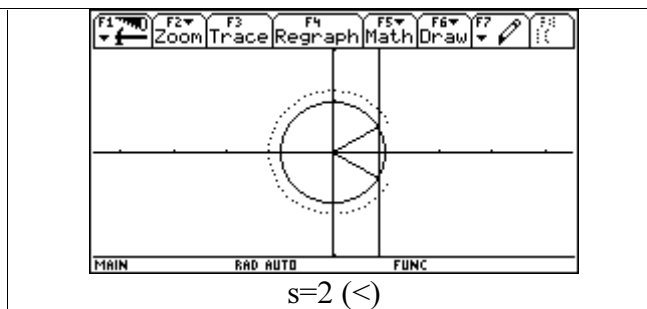
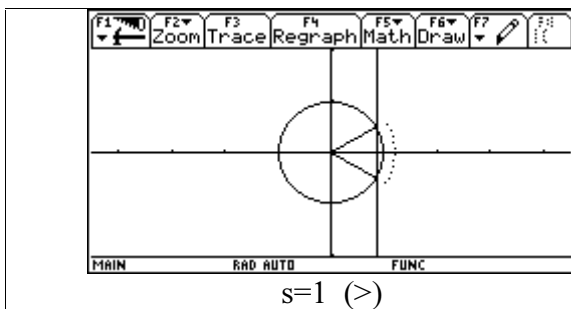
Calcola il valore dell'angolo in radianti
 Calcola l'angolo supplementare
 Converte in gradi il primo angolo
 Calcola il valore del supplementare in gradi

Scrive le soluzioni
Chiede la scelta fra $>$ e $<$ ossia : 1: per $\cos(x)>a$ o 2: per $\cos(x)<a$

Traccia un cerchio di centro $(0;0)$ e raggio 1
 Traccia una linea ad altezza $= a$
 Traccia i due segmenti che uniscono il centro delle coordinate ai due punti che rappresentano le soluzioni.
 Blocco che traccia le soluzioni della disequazione



```
discos(a)
Prgm
ClrIO
ClrDraw
Disp "cos(x)="+string(a)
If -1 ≤ a and 1 ≥ a Then
Disp "soluzioni: "
approx(cos-1(a)) → x1
-x1 → x2
x1*180/π → gr1
-gr1 → gr2
Disp "a1= "&string(gr1)&"k 360"
Disp "a2= "&string(gr2)&"k 360"
input "1: > 2: < ",s
-4.5 → xmin
4.5 → xmax
-2 → ymin
2 → ymax
Circle 0,0,1
Line a,ymin,a,ymax
Line 0,0,cos(x1),sin(x1)
Line 0,0,cos(x2),sin(x2)
If s=1 then
x2 → h1
x1 → h2
else
x1 → h1
2*π+x2 → h2
endif
For i,h1,h2,0.1
PtOn 1.2*cos(i),1.2*sin(i)
EndFor
Else
Disp "impossibile"
EndIf
EndPrgm
```

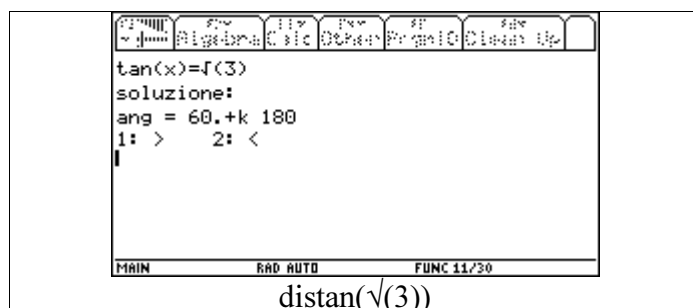


Programma `distan(a)`

Il significato delle istruzioni è come sopra

Traccia la linea Tangente al cerchio goniometrico
 Traccia la linea che unisce il punto sulla retta tangente
 (1;a) alle soluzioni sul cerchio goniometrico

Blocco che traccia le soluzioni della disequazione



```

eqtan(a)
Prgm
ClrIO
ClrDraw
Disp "tan(x)="+string(a)
Disp
Disp "soluzione: "
approx(tan-1(a))→x1
x1*180/π→gr1
Disp "ang = "&string(gr1)&"+k 180"

input "1: > 2: < ",s

-4.5→xmin
4.5→xmax
-2→ymin
2→ymax
Circle 0,0,1
Line 1,ymin,1,ymax
Line 1,a,-cos(x1),-sin(x1)

If s= 1 then
For i,x1,π/2,0.1
PtOn 1.2*cos(i),1.2*sin(i)
EndFor
For i,x1+π,3/2+π,0.1
PtOn 1.2*cos(i),1.2*sin(i)
EndFor

Else
For i,-π/2,x1,0.1
PtOn 1.2*cos(i),1.2*sin(i)
EndFor
For i,π/2,x1+π,0.1
PtOn 1.2*cos(i),1.2*sin(i)
EndFor

```

	<p>Endlf</p>
<p>s = 1 (>)</p>	<p>EndPrgm</p>
	<p>s=2 (<)</p>

Applicazioni delle funzioni goniometriche

Attività 1: Un proiettile, lanciato da terra con una velocità iniziale di v_0 m/sec con un angolo di elevazione a , si sposta orizzontalmente di moto uniforme $x = v_{0x} \cdot t$ e, verticalmente, di moto uniformemente accelerato $y = v_{0y} \cdot t - 9,8/2 t^2$, essendo $v_{0x} = v_0 \cdot \cos(a)$ e $v_{0y} = v_0 \cdot \sin(a)$.

Le equazioni parametriche del moto, in funzione del parametro t , sono quindi:

$$\begin{cases} x = v_0 \cdot \cos(a) \cdot t \\ y = v_0 \cdot \sin(a) \cdot t - \frac{9,8}{2} t^2 \end{cases}$$

La componente verticale del moto è rappresentata da un'equazione di 2° grado, cioè dall'equazione di una parabola ad asse verticale, quindi l'altezza massima corrisponde all'ordinata del vertice.

Dunque, l'altezza massima verrà raggiunta al tempo $t_m = v_{0y}/9,8$ e tale altezza massima sarà $y_m = v_{0y} t_m - 9,8/2 t_m^2$. La distanza massima, gittata, verrà raggiunta in un tempo doppio di quello dell'altezza massima e la distanza massima sarà quindi $= v_{0x} \cdot 2 \cdot t_m$.

Si vuole scrivere un programma che, inseriti velocità iniziale v_0 e angolo a , converta la misura dell'angolo a in quella in radianti al e calcoli:

- l'istante in cui viene raggiunta l'altezza massima
- il valore di tale altezza
- la gittata.

Volendo poi rappresentare il moto graficamente, si deve trovare il luogo dei punti di coordinate $(x;y)$ e quindi occorre eliminare il parametro t dalle equazioni del moto e si ottiene:

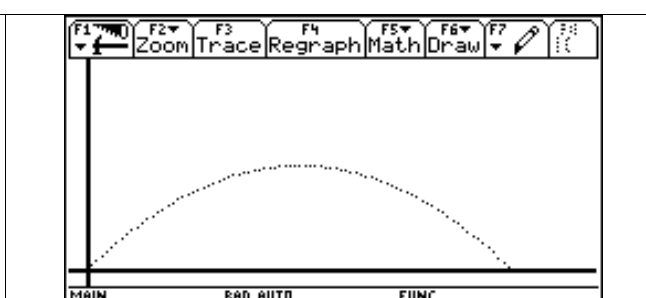
$$y = x \cdot \tan(al) - \frac{9,8}{2} \frac{x^2}{(v_0 \cdot \cos(al))^2}$$

La rappresentazione grafica del moto si ottiene facendo variare x da 0 al valore della gittata.

Un possibile risultato è il seguente:

Definisce le coordinate dello schermo grafico	<pre> proiet() Prgm ClrIO ClrDraw -10 → xmin 300 → xmax -10 → ymin 150 → ymax </pre>
Inserimento dei dati	<pre> Disp "moto di un proiettile" Input "velocita' iniz. (m/sec)",v0 Input "angolo di lancio (gradi)",a </pre>
Conversione in radianti	<pre> a/180*π → al v0*cos(al) → vx v0*sin(al) → vy vy/(9.8) → xm xm*2 → xt -9.8/2*xm^2+vy*xm → ym Disp "h max = "&string(ym)&" m al tempo </pre>

<p>Scrittura dei risultati</p> <p>Ciclo per la rappresentazione grafica : per la x è stata usata la variabile i e il passo è di 3 metri</p>	<pre> Disp "h max = "&string(ym)&" m al tempo "&string(xm) &"sec" xt*vx→dm Disp "gittata "&string(dm)&"m" Disp "premi enter" Pause For i,0,dm,3 i*tan(al)-4.9*i^2/(v0*cos(al))^2 → yy PtOn i,yy EndFor EndPrgm </pre>
---	---



Attività 2

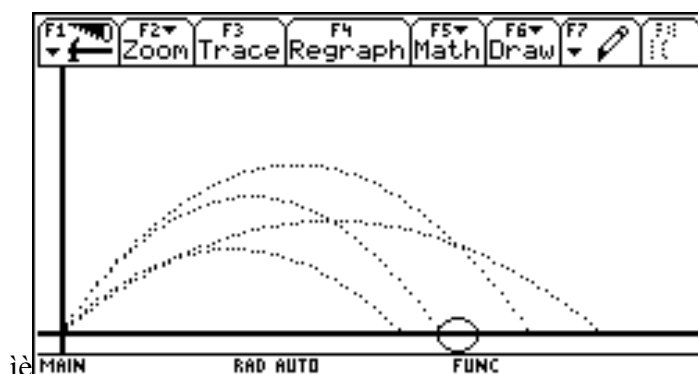
Il programma può essere modificato in un gioco. Cioè si può far generare al calcolatore un numero casuale (compreso fra 150 e 300) che corrisponde all'ascissa di un bersaglio collocato al suolo e quindi innescare un loop che termina quando il proiettile colpisce il bersaglio.

Un possibile risultato è il seguente.

Può essere ottenuto modificando il programma precedente con l'inserimento delle istruzioni incorniciate.

<p>Disegna il bersaglio (cerchio di raggio 10)</p> <p>Inizializza a zero il contatore dei tentativi</p> <p>Inizia il Loop</p> <p>Incrementa il contatore dei tentativi</p>	<pre> proiet2() Prgm ClrIO ClrDraw rand(150)+150 → xb -10 → xmin 300 → xmax -10 → ymin 150 → ymax Circle xb,0,10 0 → k Disp "tiro a bersaglio" Loop k+1 → k Input "velocita' iniz. (m/sec)",v0 Input "angolo di lancio (gradi)",a a/180*π → al v0*cos(al) → vx </pre>
--	---

<p>Inizia il grafico del moto</p> <p>Controlla se il bersaglio è stato colpito</p> <p>Uscita se il controllo è positivo</p> <p>Fine del ciclo</p> <p>Comunica il numero dei tentativi</p>	<pre> v0*sin(al) →vy vy/(9.8) → xm xm*2→ xt - 9.8/2*xm^2+vy*xm → ym Disp "h max = "&string(ym)&" m al tempo "&string(xm) &"sec" xt*vx→dm Disp "gittata "&string(dm)&"m" Disp "premi enter" Pause For i,0,dm,3 i*tan(al)-4.9*i^2/(v0*cos(al))^ → yy PtOn i,yy EndFor If abs(i-xb)<10 then Exit EndLoop Disp "centro in "&string(k)&" tentativi" EndPrgm </pre>
---	--



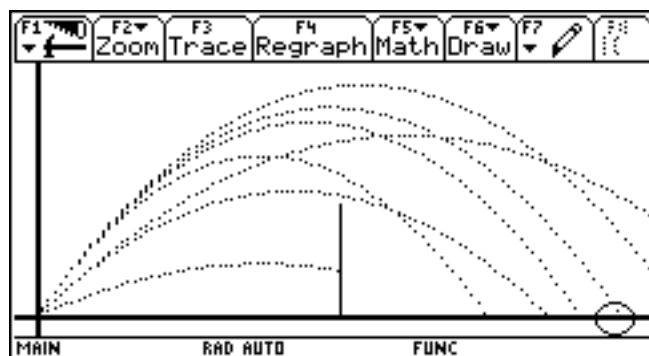
Esercizio:

Una ulteriore modifica del programma, per complicare un po' il gioco, consiste nel far generare al calcolatore uno sbarramento verticale.

Soluzione: Basta aggiungere al programma precedente le istruzioni incorniciate.

<p>Genera l'ascissa del bersaglio</p> <p>Genera l'ascissa dello sbarramento</p> <p>Genera l'altezza dello sbarramento</p>	<pre> Proiet3() Prgm ClrO ClrDraw rand(150)+150 → xb rand(50)+xb/2 → xs rand(80)+30 → ys -10 → xmin 300 → xmax -10 → ymin 150 → ymax Circle xb,0,10 0 → k </pre>
---	--

	<pre> 0→k Disp "tiro a bersaglio" Loop k+1→k Input "velocita' iniz. (m/sec)",v0 Input "angolo di lancio (gradi)",a a/180*π → al v0*cos(al) →vx v0*sin(al) →vy vy/(9.8) → xm xm*2→ xt - 9.8/2*xm^2+vy*xm → ym Disp "h max = "&string(ym)&" m al tempo "&string(xm) &"sec" xt*vx→dm Disp "gittata "&string(dm)&"m" Disp "premi enter" Pause </pre>
<p>Traccia la linea dello sbarramento</p>	<pre> Line xs,0,xs,ys For i,0,dm,3 i*tan(al)-4.9*i^2/(v0*cos(al))^ → yy PtOn i,yy </pre>
<p>Controlla se il proiettile è prossimo allo sbarramento</p> <p>Controlla se il proiettile supera lo sbarramento</p> <p>Va alla label sbb</p>	<pre> If abs(i-xs)< 2 then If yy < ys then Disp "sbarramento non superato" Goto sbb endif endif EndFor lbl sbb </pre>
<p>Controlla se il bersaglio è stato colpito</p>	<pre> If abs(i-xb)<10 then Exit EndLoop Disp "centro in "&string(k)&" tentativi" EndPrgm </pre>



Attività 3 - Giostra

Una giostra di Mirabilandia, chiamata “Il The del Cappellaio Matto”, consiste in un piatto che gira che contiene tazze a loro volta girevoli.

Le funzioni goniometriche ci consentono di costruire un programma che rappresenta graficamente la traiettoria descritta da una persona sulla giostra, ossia seduta vicino al bordo di una tazzina.

Facciamo le ipotesi semplificative che sia il moto del piatto che quello della tazzina abbiano velocità angolare costante.

Consideriamo un piatto di raggio 8 m e supponiamo che la tazzina abbia centro a 6 m dal centro ed abbia raggio 2 m. Ad ogni passo (i) la giostra ruota di 0,1 rad e la tazzina di 0,57 rad.

Le equazioni del moto del centro della tazzina sono:
$$\begin{cases} x = 6 \cos(i) \\ y = 6 \sin(i) \end{cases}$$

Le equazioni del moto della persona, rispetto al centro della tazzina sono:
$$\begin{cases} xx = 2 \cos(j) \\ yy = 2 \sin(j) \end{cases}$$

Per cui le equazioni del moto della persona rispetto al centro del piatto sono:
$$\begin{cases} xx = 2 \cos(j) + x \\ yy = 2 \sin(j) + y \end{cases}$$

Il programma esegue due tracciati:

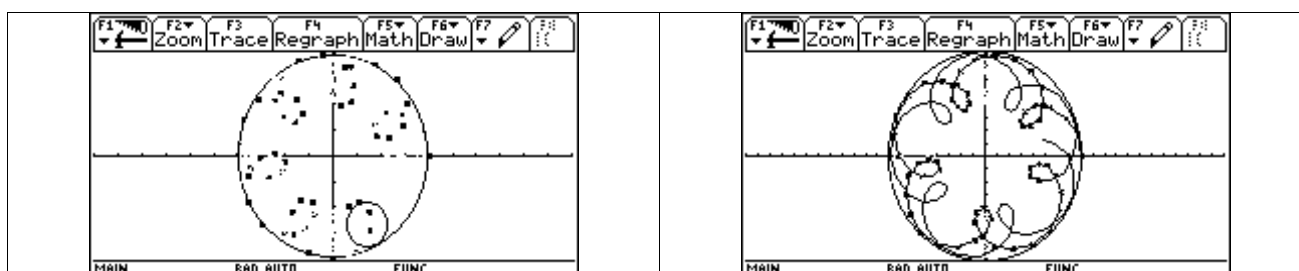
nel primo mostra le posizioni occupate di volta in volta dalla persona sulla tazzina disegnando il cerchio che rappresenta la tazzina e un cerchietto che rappresenta la persona, nel secondo, per rendere più chiara la traiettoria, non viene tracciato il cerchio della tazzina e le varie posizioni della persona vengono unite da segmenti

Vediamo il programma:

	giostra() Prgm ClrDraw
Definisce le coordinate di schermo	-20 → xmin 20 → xmax -8 → ymin 8 → ymax
Traccia il piatto della giostra	Circle 0,0,8.2
Inizializza a 0 l'angolo di rotazione della persona	0 → j
Inizia il ciclo di rotazione della tazza sul piatto	For i, 0,2*π,0.1
Coordinate del centro della tazza	6*cos(i) → x 6*sin(i) → y
Traccia la tazza	Circle x,y,2
Coordinate della persona sulla tazza	2*cos(j)+x → xx 2*sin(j)+y → yy
Cerchietto rappresentante la persona	Circle xx,yy,0.2
Incremento angolo di rotazione della persona	J + 0.57 → j
Cancellazione del cerchio tazza	Circle x,y,2,0
Fine del ciclo	EndFor
Seconda parte	4 → xx1
Coordinate del punto iniziale	0 → yy1 0 → j
Ricomincia il ciclo della tazzina	For i, 0,4*π,0.1
Ciclo analogo al precedente	6*cos(i) → x 6*sin(i) → y

Traccia la linea che unisce il nuovo punto al precedente	<pre> 2*cos(j)+x → xx 2*sin(j)+y → yy line xx,yy,xx1,yy1 J + 0.57 → j xx → xx1 yy → yy1 EndFor </pre>
Memorizzazione delle coordinate del punto	<pre> EndPrgm </pre>

Digitare in HOME : giostra() , si otterrà:



Attività 4 – La giostra dei pianeti

Sappiamo tutti che i pianeti del nostro sistema solare ruotano attorno al Sole, però, nel sistema di riferimento della Terra , il Sole ruota attorno alla Terra e trascina gli altri pianeti in questa rotazione come la tazzina trasporta i passeggeri che ruotano sulla giostra.

Il programma che segue è una modifica di quello della giostra in cui non è stato usato il For per il ciclo del Sole perché si richiedeva un passo negativo del ciclo per attuare un moto in senso orario e il calcolatore non lo consente. E' stato usato quindi il Loop.

Il sistema di riferimento è stato cambiato adeguandolo a quello del Sistema solare. Ricordiamo alcuni dati:

Raggio dell'orbita terrestre = 150 milioni di Km Periodo del moto del Sole = 365 giorni

Raggio dell'orbita di Mercurio = 57,9 milioni di Km Periodo di Mercurio = 88 giorni

Raggio dell'orbita di Venere = 108 milioni di Km Periodo di Venere = 225 giorni

Raggio dell'orbita di Marte = 228 milioni di Km Periodo di Marte = 687 giorni

E' stato quindi definito come sistema di riferimento :

$x_{min} = -500$, $x_{max} = 500$, $y_{min} = -250$, $y_{max} = 250$

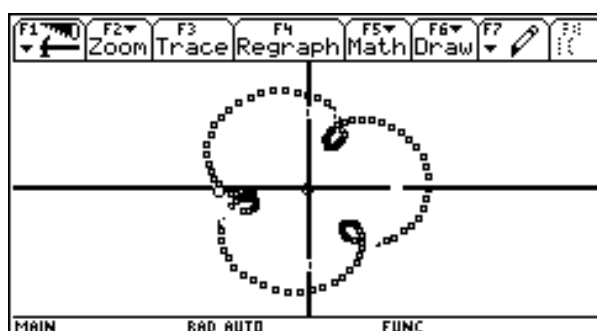
affinchè fosse possibile rappresentare i moti dei tre pianeti.

Per rappresentare il grafico giorno per giorno sarebbe stato necessario definire il passo $1/365$ per $1/pp$ per i pianeti, ma, per velocizzare l'esecuzione questi incrementi sono stati moltiplicati per 20.

Definisce il sistema di coordinate di schermo	<pre> Pianeti() Prgm ClrDraw -500 → xmin 500 → xmax -250 → ymin 250 → ymax </pre>
---	--

Parametri del pianeta (in questo caso Mercurio)	88 → pp 57.9 → rp @ pp = periodo pianeta @ rp = raggio orbita pianeta @ mercurio : pp=88 rp= 57.9 @ venere : pp=225 rp = 108 @ marte : pp= 687 rp = 228
Cerchio che rappresenta la Terra Inizializzazione i e j	Circle 0,0,10 0 → j π → i
Inizio del ciclo Coordinate del Sole	Loop $150 \cdot \cos(i)$ → x $150 \cdot \sin(i)$ → y
Cerchio che rappresenta il Sole	Circle x,y,10
Coordinate del pianeta	$rp \cdot \cos(j) + x$ → xx $rp \cdot \sin(j) + y$ → yy Circle xx,yy,5
Cerchio che rappresenta il pianeta Incremento di j (1/periodo * 20) Incremento di i (1/periodo * 20) Cancellazione del cerchio del Sole Controllo di uscita dal loop	j-1/pp*20 → j i-1/365*20 → i Circle x,y,10,0 If i < - π then Exit Endif EndLoop
Ridisegna il Sole	Circle x,y,10 EndPrg

Digitando il HOME : pianeti() si ottiene:



Disegnare le orbite di Venere e Marte modificando rp e pp .