LE MOTEUR DE JEU

Auteur : Jean-Baptiste Civet

TI-83 Premium CE

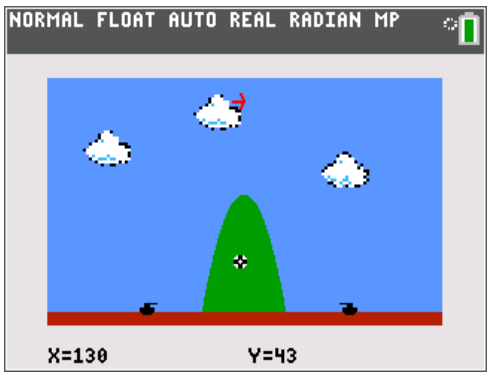
Mots-clés : représentation graphique, programmation graphique.

Fichiers associés : JeuDeTir_fiche3_jeu_eleve.pdf, DINIT.8XP, DMONTAGN.8XP, DSPLASH.8XP, DTANK.8XP, DVENT.8XP, DBOOM.8XP, DTRAJECT.8XP, DTROU.8XP, JOUEUR1.8XP, JOUEUR2.8XP, TANK.8XP

1. Objectifs

- Construire le moteur de notre jeu (interaction et tir), en utilisant les primitives graphiques spécifiques à la TI-83 Premium CE dans l'environnement de programmation de la calculatrice.
- La trajectoire de tir sera parabolique et sera impactée par la direction et la force du vent.
- Ce projet sera l'occasion de comprendre l'usage des variables globales dans l'environnement de programmation.
- Il s'agit de la troisième activité d'un ensemble de trois activités aboutissant à la programmation d'un jeu fonctionnel.

2. Énoncé



Maintenant que le décor est construit et les variables prêtes, nous allons, dans cette dernière activité, mettre en place les éléments d'interaction permettant de configurer le tir, le représenter et déterminer s'il s'agit ou non d'un tir gagnant.

3. Commentaires

Le moteur de jeu consistera en un seul programme appelant l'ensemble des programmes créés.

Le programme principal initialisera, après l'affichage du splashscreen, le dessin de la montagne, des tanks et du vent (reprise du travail précédent).

Cette fois-ci l'enjeu technique se situe dans la saisie de l'angle et de la force du tir par le joueur qui a la main.

Le dessin de la trajectoire de tir ne posera pas de problème particulier. Il faudra correctement placer le départ du tir en fonction de la position du canon.

Les équations ont été données aux élèves. Là encore, l'enseignant réalisera ses arbitrages dans le scénario qu'il souhaite leur proposer. Un dernier travail, assez logique, sera mené pour déterminer si le tir est gagnant ou non.

4. Conduite de l'activité

Comme toujours, la lecture attentive des programmes est indispensable (voir fichiers associés plus haut).

Afin de pouvoir faire communiquer correctement les programmes entre eux, le respect des variables « réservées » et des noms de programme est indispensable.

Le programme principal est Tank.8XP

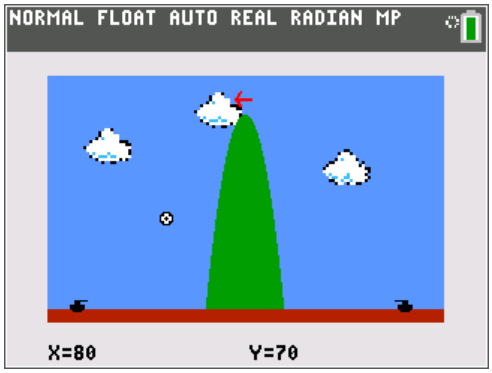
Interception des saisies

Nous avons généralement l'habitude d'utiliser la fonction « Input » en la faisant suivre d'un nom de variable.

Dans notre programme, nous souhaitons obtenir deux valeurs : l'angle de tir et sa force. Il était tout à fait possible de saisir dans notre programme :

```
Input "Angle : ",A
Input "Force : ",F
```

Cette méthode tout à fait valable, nous amène à quitter l'écran graphique pour retourner dans l'écran de saisie.



Il se trouve qu'un appel sans argument de la fonction « Input » nous permet de saisir simultanément deux valeurs dans les variables X et Y, simplement en déplaçant le curseur sur l'écran et en lisant les valeurs dans l'écran graphique (bord inférieur de la fenêtre).

On peut voir sur l'image ci-contre le curseur à l'écran. Ainsi, le joueur s'apprête à valider un angle de 80° pour une vitesse de 70 m/s.

Nous préférons donc le programme suivant :

```
Input
X°→A
Y→F
prgmDTRAJECT
If G=0
prgmDTRAJECT
```

Par défaut, nous forçons le mode radian dans le programme DInit. L'ajout de ° à la variable X permet de forcer la calculatrice à interpréter l'angle en degré.

Nous traçons une première fois le tir.

S'il n'est pas gagnant (G = 0, voir annexe), on relance le programme à l'identique. Cela aura pour effet d'effacer le tir.

La trajectoire des tirs

Pour démarrer la programmation de cette activité, il faut préalablement résoudre la question de la trajectoire du tir. On a donné aux élèves les équations suivantes exprimées en fonction du temps :

$x(T) = (E+F\cos(A))T$

$y(T) = F\sin(A)T-5T^2$

On souhaite que le tir parte de l'extrémité du canon. Il faudra tenir compte du joueur qui a la main.

Pour cela, on propose la solution suivante :

$x(T) = (E+JF\cos(A))T+D$

$y(T) = F\sin(A)T-5T^2+15$

En effet, les canons des deux chars sont tous à la même hauteur, soit 15 pixels.

C'est au moment de l'affectation de la variable D (voir annexe) que nous ajusterons l'abscisse du tir. Nous le ferons dans le corps du programme principal Tank.8XP

Nous rappelons que la variable J peut prendre 1 (joueur de gauche) ou -1 (joueur de droite) comme valeur. Ainsi, si le joueur 1 a la main, on ajoute le produit Fcos(A), si le joueur 2 a la main, on retranche ce produit. Cette méthode permet d'orienter la trajectoire de la gauche vers la droite (joueur 1) ou de la droite vers la gauche (joueur 2).

Dessin de la trajectoire des tirs

Nous avons recours à la fonction « Pt-Change(X,Y,Couleur) ».

Cette fonction allume un pixel aux coordonnées (X;Y) selon le paramètre « Couleur ». Deux possibilités, le pixel était « éteint » et la procédure est réalisée. Le pixel était déjà allumé, dans ce cas, la procédure éteint le pixel, laissant alors apparaitre le pixel de l'image de fond.

Cette fonction est donc intéressante car elle n'interagit qu'avec l'écran graphique et ne touche pas à l'image de fond. Ainsi le ciel et ses nuages resteront intacts, ce qui ne sera pas le cas de la flèche du vent lorsque son dessin sera touché par le dessin de la trajectoire du tir.

De plus, en appelant deux fois de suite le programme de dessin, on peut effacer la trajectoire de tir et passer la main au joueur suivant.

Ce qui donne, par exemple, le code suivant pour le dessin de la trajectoire :

```
0→T
0→S

While S=0
        T+0.2→T
        (E+JFcos(A))T+D→X
        FTsin(A)–5T²+15→Y
        Pt-Change(X,Y,BLUE)
                If X<O–4 or X>O+4
                Then
                        If Y<Y₁ or Y≤9
                        1→S
                Else
                        If Y<15
                        Then
                                1→G
                                1→S
                        End
                End
End
prgmDTROU
```

On initialise le temps T et la boucle S.

On commence le tracé en incrémentant le temps et en calculant les coordonnées de la balle.

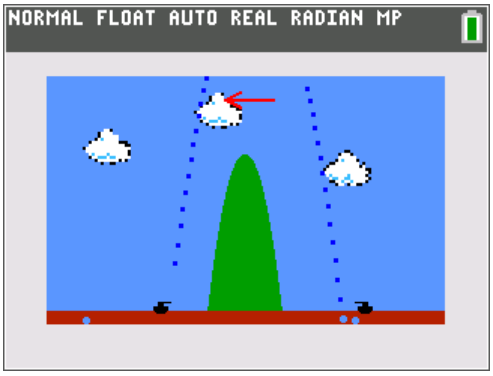
On dessine ou efface la position.

On teste la position par rapport à l'objectif O.
Si on est avant ou après l'objectif, on vérifie si on a touché la montagne ou le sol et si oui on s'arrête (S = 1)….

Sinon si on est dans les environs de l'objectif, on vérifie si on est au-dessus ou pas…

Si on n'est pas au-dessus, on a gagné et on s'arrête.

Dans tous les cas, on marque l'impact au sol ou sur la montagne ou sur le tank en appelant le programme DTrou.



Sur l'image ci-contre, on voit la trajectoire du tir en train d'être effacée. Les pixels sont remplacés par la couleur du fond.

Le moteur du jeu

Nous avons volontairement décomposé le projet en un nombre important de sous-programmes.

Cela permet de gagner en lisibilité d'une part. D'autre part, nous verrons que cela ouvre des possibilités de travail supplémentaire.

Voici le moteur de notre jeu :

prgmDSPLASH	On lance l'écran d'introduction.
Pause	On attend l'appui sur la touche « Entrer »
prgmDMONTAGN	On dessine la montagne.
132–√((H–9)/L)–13→P	On calcule la place disponible.
1+int(Prand)→X	On positionne le premier char d'assaut.
1→J	On définit qu'il s'agit du joueur de gauche.
X→U	On mémorise sa position (variable réservée U).
prgmDTANK	On le dessine.
255–int(Prand)→X	On positionne le deuxième char d'assaut.
-1→J	On définit qu'il s'agit du joueur de droite.
X→V	On mémorise sa position (variable réservée V).
prgmDTANK	On le dessine.
prgmDVENT	On calcule et dessine la vitesse du vent.
0→G	On initialise la partie.
While G=0	Tant que personne n'a gagné…(la boucle de jeu)
U+13→D V+7→O	On définit le départ du tir du joueur de gauche, sa position+13 pour l'extrémité du canon et le milieu du joueur de droite (position + 7) pour l'objectif.
1→J	Le joueur de gauche a la main.
prgmJOUEUR1	S'il n'a pas atteint son objectif, on passe la main au joueur2.
If G=0 Then V→D U+7→O -1→J	On définit la nouvelle position de départ de tir (ici canon et position sont confondus) et la position de l'objectif (milieu du joueur de gauche).
prgmJOUEUR2	
End	Fin de la boucle While
End	Quelqu'un a gagné, on lance donc la procédure de fin.
prgmDFINAL	

5. Conclusion

Au cours de ce projet, on a vu, dans chacune des trois activités proposées, des aspects techniques et mathématiques. A l'issu de ce travail, on a acquis des bases solides de programmation graphique sous TI-83 Premium CE et ce, à l'aide de nos indispensables connaissances mathématiques.

Pour l'enseignant, il y a probablement 3 façons de faire en sorte que les élèves s'approprient un tel projet. Tout d'abord, par production, selon un cahier des charges donné. C'est dans cet esprit que les fiches élèves ont été construites. Et c'est pour cela que l'on a multiplié les sous-routines. Ainsi on peut demander aux élèves de produire l'intégralité du programme de dessin de la montagne voire du décor dans son ensemble que l'on rebranche ensuite sur le moteur de jeu.

Une autre façon peut être de demander aux élèves d'étudier le code source (en les guidant éventuellement par un jeu de questions) puis de produire un projet de jeu similaire. Par exemple, un jeu de lancer franc au basketball. Il s'agira notamment de redessiner un décor adapté et de comprendre l'influence sur le code du

changement de position du départ de ballon et de son arrivée, les équations pouvant être reprises en l'état si le présent projet a bien été compris.

Enfin, une dernière façon, consiste en l'enrichissement du projet. Ainsi les élèves auront peut-être envie de s'emparer davantage encore de ce projet, grand classique de la programmation, en implémentant de nouvelles fonctions comme un tableau de score (gérant par exemple le nombre de coups utilisés pour atteindre la cible et le nom des joueurs incluant un système de « HighScore » par l'utilisation des listes) ou bien encore en donnant une « intelligence » au joueur n°2 contre qui, ils pourront se mesurer par la suite. Se mesurer à la machine est toujours un challenge palpitant…le programmer encore davantage. Nous en reparlerons dans des projets futurs.

6. ANNEXE

• En vue de la poursuite de l'ensemble du projet, respecter les affectations de variables :

A : **A**ngle de tir sélectionné par un joueur

D : **D**épart (position de) du tir

E : vitesse du vent (valeur comprise entre -5 et 5) (la variable V est déjà attribuée)

F : **F**orce (vitesse) initiale du tir

G : **G**agné ? (1 si oui, 0 sinon)

H : **H**auteur de la montagne (valeur comprise entre 60 et 160)

J : **J**oueur en action (1 si joueur de gauche, -1 si joueur de droite)

L : **L**argeur de la montagne (valeur, par exemple comprise, entre 0,125 et 0,5)

O : **O**bjectif (position de l')

P : **P**lace disponible

T : **T**emps pour le calcul de la trajectoire du tir

U : tank de gauche

V : tank de droite

Y1 : équation de la montagne

• Respecter de même les noms de programmes suivants :

DInit.8XP : le programme de préparation de l'écran graphique.

DMontagn.8XP : le programme de dessin de la montagne.

DSplash.8XP : le programme de dessin de l'écran d'introduction du jeu.

DTank.8XP : le programme de dessin des chars d'assaut.

DVent.8XP : le programme de dessin de la force du vent.

DBoom.8XP : le programme qui dessine l'explosion du char d'assaut.

DTraject.8XP : le programme qui dessine la trajectoire du tir.

Joueur1.8XP : le programme qui gère la saisie de l'angle et de la force du joueur1.

Joueur2.8XP : le programme qui gère la saisie de l'angle et de la force du joueur2.

Tank.8XP : le moteur de jeu gérant l'ensemble des appels et procédures.