Moteur de Jeu personnel de plateforme pour MMF2

Auteur David Newton
Traduction : Patrick Gimeno

Document en cours de traduction

Contents

Sommaire

A Basic Fastlooping Platform Movement

Once you've made your first few projects in MMF, one of the most important steps is being able to write your own custom movements through the Event Editor rather than relying on the default ones. The default platform movement is one of the most infamous elements of the Click range of software for its distinctive characteristics, including resetting your horizontal movement to 0 when you land from a jump, and becoming stuck in the walls or ceilings of the frame.
The most basic custom platform movements (those that don't use the fastloop functionality) go some way to getting around these problems, but introduce new ones as well - without using the default movements, you have to concentrate a lot more on your own collision detection, and if you're moving an object more than one pixel per frame you have to work around the danger of your object getting stuck in the obstacle backdrop because of moving too far in between checks.
This is where the idea of fastloops comes in. Normally, the event list is read from top to bottom, with any actions being carried out in order. After a read-through of the event list, the screen is

updated and the process starts again. Fastloops, as their name implies, can be run many times in one cycle of the event list. The action "Start loop" under the "Special" object is used to trigger a fastloop, and on each loop, events that begin with the condition "On loop (name of the loop being run)" are run in order, as many times as are specified by you.
If none of this is making any sense, keep going - it does get better. The best way to learn how to use them is by example.

Un mouvement de plateforme simple utilisant les boucles rapide.

Après avoir créer quelques premiers projet dans mmf, une des choses les plus importantes est le fait d'écrire son propre type de mouvement au travers de l'event editor à la place de ceux proposé par défaut.
En effet, le mouvement de plateforme par défaut est l'une des plus (... commentaire sans intérêt) pour ses caractéristiques distinctes, incluant par exemple le "reset" du mouvement horizontal à 0 quand tu attéri d'un saut, et devient de la colle dans les murs ou les plafonds. (l'objet se retrouve bloqué)
Le plus basique mouvement de platformes customisé (par soi meme) (qui n'utiliserait pas la fonctionalité fastloop) ira tout droit vers ce genre de problème, mais en introduit de nouveaux.

Contrainte :
En n'utilisant pas le mouvement par défaut, vous devez vous concentrer bien plus sur vos propre détections de collisions.
Si par exemple vous bougez un objet sur plus d'un pixel par "frame" ou "image par seconde" vous devez travailler sur le danger qu'un objet puisse se "coller" à l'intérieur d'un obstacle parce qu'il a pu bouger trop loin entre 2 détections. (phénomène de colle)

C'est là ou l'idée des fastloop(boucles rapides) rentre en jeu.

Fonctionnement :
Normalement la liste des "evènements" est "lue" par l'ordinateur du haut vers le bas avec toutes les actions réalisées ds l'ordre . Après que tout soit lu, l'écran est effacé et mis à jour et tout recommence.
"Fast loops" ou "boucles rapide" comme leur nom l'indique, peut etre lancé, (joué) autant de fois que l'on veut dans un cycle de la liste d'évènement. >> dans un soucis de précision.

Comment l'utiliser :
"Lancer boucle rapide" est là pour déclencher une boucle rapide, par exemple >> l'évènement qui commence par "on loop" (MaBoucle) appelle la boucle rapide et l'exécute ds l'ordre et autant de fois que spécifié par le créateur.

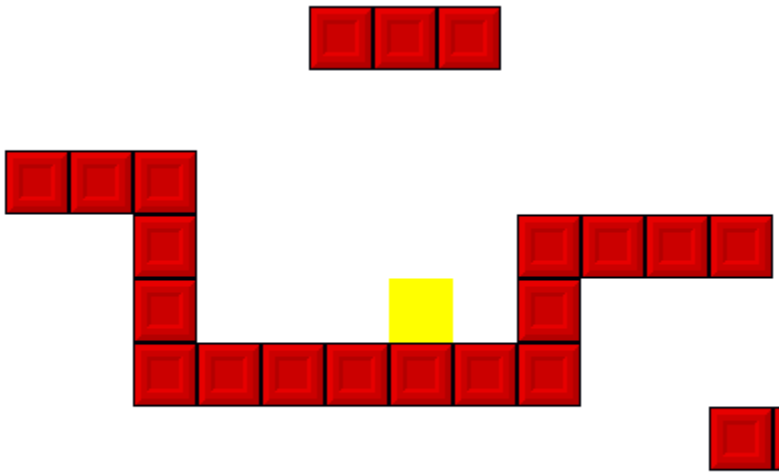
A Basic Fastlooping Platform Movement

The Basics

To start off with, you'll need a player object to control and some obstacles to collide with. For the moment, just keep these as simple rectangles/squares - the actual artwork can be added later on. You don't need to add any sort of movement to the Player object, as with this custom engine, we'll be controlling its position on the screen entirely using events.

Un mouvement de plateforme simple à base de Boucles rapide.

Pour commencer, vous avez besoin d'un objet qui sera votre "joueur" à contrôler, et des obstacles pour entrer en collision avec.
Pour le moment, créer les aussi simple que des carrés / rectangles.
Des graphismes seront ajoutés par la suite.
Vous n'avez pas besoin d'ajouter de mouvement par défaut à votre objet "joueur", comme un moteur de jeu customisé (à votre sauce) nous allons controler la position du joueur entièrement avec des évènements.



Jumps and Gravity

The first consideration in a custom platform movement is handling how the player object is affected by gravity. To perform a convincing jump, the player should start off moving quickly upwards, then gradually slow down and begin moving downwards again after reaching the top of an arc. Therefore, the gravity value being applied to the player should start off with a negative value, and be gradually increased until the player is once again standing on solid ground.
Add a new alterable value to the player object and name it "Grav" (as "Gravity" is a reserved word in MMF2). We're going to use this value to store the number of pixels that the player should move per event cycle due to the effects of gravity. It should start off at 0, so leave the default value alone.

Saut et gravité

La première considération dans un moteur personnel est de définir comment l'objet joueur est affecté par la gravité. Pour réussir un saut convaincant, le joueur doit bouger rapidement vers le haut, puis ralentir tout en haut et retomber plus lentement pour accélérer du a l'attraction terrestre. Par conséquent, la valeur numérique de la gravité commencera de facon négative et s'incrémentera jusqu'a que le joueur est atteint une plateforme.
Ajouter une valeur modifiable au joueur et appeler la Grav (gravité est un mot réservé dans mmf2). On va utiliser cette valeur pour stoker le NOMBRE DE PIXELS que le joueur pourra bouger par cycle d'évènements du à l'effet de la gravité
Nous devons commencer à 0. Donc laisser la valeur a zero.

Gravity should continually accelerate the player object down the way (unless it's already safely on top of an obstacle backdrop). Start a new event group if you want to keep things a little neater, and add the event:
+ Always
-> Add 1 to Grav("Player")

La gravité doit continuellement accélérer l'objet joueur vers le bas. (sauf s'il est déja en sécurité au dessus d'un obstacle).
Démarrer un nouveau groupe d 'évènements si vous voulez garder les choses un peu plus claire et ajouter l'évènement suivant :

+ Always
-> Add 1 to Grav("Player")

We're not actually going to move the player directly using this value. Instead, we're going to use a fastloop to handle the player's movement due to gravity. The set of events we need to run in our fastloop will continually check to see whether the player can move any further up or down, and if so, move the player by one pixel. This process will be repeated as many times as is needed, so that the player passes through all the intermediate points without the screen being updated. This is different from an engine without fastloops that just moves the player object the required number of pixels at once.
Add the events:
+ On loop "gravity"
+ Grav("Player") > 0
-> Set Y position of Player to Y("Player")+1

Nous n'allons pas bouger le joueur directement en utilisant cette valeur. Au lieu de cela nous utilisons un fastloop(boucle rapide) pour gérer le mouvement du joueur du a la gravité.
La définition des évènements que nous avons besoin d'exécuter dans "la boucle rapide" continuera de vérifier si le joueur peut bouger en haut ou en bas, et si oui déplace le joueur d'1 pixel.
Ce processus sera répété à chaque fois que ce sera nécessaire, ainsi le joueur passera a travers tous les points intermédiaires sans que l'écran soit rafraichi.
Voila la différence avec un moteur sans boucle rapide qui bouge le joueur du nombre de pixels requis en une fois.

Ajouter les évènement suivant :
+ On loop "gravity"
+ Grav("Player") > 0
-> Set Y position of Player to Y("Player")+1
+ On loop "gravity"
+ Grav("Player") < 0
-> Set Y position of Player to Y("Player")-1

These events look at whether the player should move up or down, and moves the player object by 1 pixel in the correct direction.

Ces évènements regardent si le joueur peut bouger en haut ou en bas, et bouge le joueur d'1 pixel dans la direction voulue.

+ On loop "gravity"

+ Player is overlapping an obstacle background
+ Grav("Player") > 0
-> Set Y position of Player to Y("Player")-1
-> Set Grav("Player") to 0
-> Stop loop "gravity"

+ On loop "gravity"
+ Player is overlapping an obstacle background
+ Grav("Player") < 0
-> Set Y position of Player to Y("Player")+1
-> Set Grav("Player") to 0
-> Stop loop "gravity"

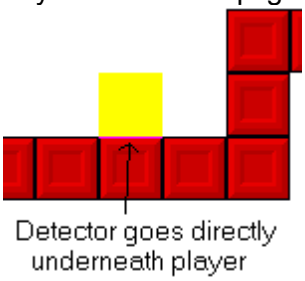
These events detect if the player has moved into a backdrop, and move the object back a pixel. The first event moves the object up again if it's moved into the ground, and the second moves it down if it's in a ceiling. When the player has moved into a backdrop, we no longer want him to keep rising or falling, so the gravity is set to 0 and we stop any further gravity loops from happening on this event cycle.
Now that the events are in place to handle the gravity, we need to allow the check to be triggered. When the player is on the ground, nothing should happen, but as soon as the gravity is not 0, we want to start the loop that we've just made.

Ces évènements détectent si le joueur s'est déplacé dans un décor, et bouge l'objet en arriere d'1 pixel
Le premier évènement bouge l'objet vers le haut encore s'il a bougé ds le sol et vers le bas si c'est un plafond.
Quand le joueur a bougé dans un décor, nous ne voulons pas qu'il continu de monter ou descendre, donc la gravité est définie a 0, et stoppe toute boucle de gravité dans ce cycle d'évènement.
Maintenant que les évènements sont en place pour gérer la gravité, nous avons besoin de vérifier s'il doit être activé.
Quand le joueur est sur le sol, rien ne peut arriver, mais dès que la gravité est différente de zero, nous voulons démarrer la boucle rapide que nous avons fabriqué.

+ Grav("Player") <> 0
-> Start loop "gravity" ABS(Grav("Player")) times.

This event will start the "gravity" loop whenever the gravity is not zero. When the loop is triggered, all events that begin with "On loop 'gravity'" will be run the amount of times we specify. Each loop will move the player by only one pixel, so we want to loop for as many times as the gravity value says we should. The ABS() is there so that we always have a positive number of loops - giving a negative number will cause the application to crash.
The last event to add here covers more familiar ground...
+ Player presses Fire 1
-> Set Grav("Player") to -10
When the player presses the jump trigger, the value of the gravity should be set to a negative number, so the gravity loop takes the player up the screen. Gravity is continually subtracted from by the first event we made, so eventually the gravity will become positive and the player will fall back down again.
Now's the time to start the application up and see the effects of these events. When Fire 1 is

pressed, the player object will jump up and fall back again, never moving too far into the floor because we check for this happening in the fastloop. Also, try putting a ceiling above the player and bashing into it using the jump - you'll be able to see the effects of the event that stops the player rising into ceilings.
You'll notice that you're able to jump in mid-air as we haven't checked for the player actually being on the ground yet - but we can add this check in. There are multiple ways to do this, but to keep the event list simple at this point, we're going to add a detector object that always sits below the player, so we can test whether it's overlapping the backdrop or not.
Create a new active object, the same width as your player and 1 pixel high (you only need it to be one pixel as we're handling the movement using the fastloop method - there's no danger of it "missing" a backdrop.) As the player object will never actually be overlapping a backdrop outside of the fastloop that moves it, we need to position this object directly underneath the player so that it will be overlapping the backdrop if the player is on top of a platform.
Name the object "Detector". This object would usually be made invisible at runtime, but we want to see its effects clearly for now, so leave it visible and go into the Event Editor. Add an action to any of the "On loop 'gravity'" events to position the detector directly below your player.



Now copy the action down into the other "On loop" events - whenever a loop event moves the player object. This is the catch of using fastloops - you have to remember to update everything necessary on each loop, otherwise your engine may have problems - and, because the loops are performed before the frame is updated, it won't be obvious where they are.
This detector has to be checked against whenever the player presses the Jump key. It's as simple as it sounds - just change the existing "Player presses Fire 1" event to this:
+ Player presses Fire 1
+ Detector is overlapping a backdrop *[NEW]*
-> Set Grav("Player") to -10
This only allows the jump to take place if the player is standing on a backdrop (and therefore the detector is overlapping it). Start the application up again and see its effects.