2.4 Events and actions

A game, when running or being played, is a dynamic system. It is something that changes, often quickly from the player point of view. These constant change in the game is actually the result of small very frequent changes, perceived as a greater, more complex turn of events. The key to determine how the game evolves is to understand the basics changes and how establish them from their components called "events" and "actions"

To better understand events and actions, we must first consider that a computer game is actually a program which is running continuously. Sometimes this program is known as "game engine". This program is frequently monitoring the keyboard, mouse, and other controllers; it can read the time, knows the position and direction of any object from the game, etc. Therefore, the game engine is the first one to know that something has happened: either a key has been pressed, or a delay has been elapsed, or two objects have touched in the screen, etc. GameMaker calls events to these things that occur and can affect the game. Many events come up as a result of the player interacting with the computer. Some other events occur as a result of the evolution of the game.

The game engine controls the game and may change the state of anything in it, at any desired moment. It can reposition or move an object, add some damage to it, create a new object, etc. These changes happens as intended by the programming done by the developer, which the engine game is only the executor of the programm. GameMaker calls "actions" to a change which is specified by the developer. The key here is that every action happens as a result of an event, even actions which are set to happen randomly are linked to events with a random component on them. The goal of the developer is to set which actions happens as a consequence of which events. We'll analyze events in depth first and then we'll see how actions are linked to them

There are a long list of pre-established events by Game Maker, but the game developer is free to take any action in response to any chosen event.

The following list introduces the most important events:

- KeyDown, KeyPressed, KeyUp. (These are the names of the event for any key, not for the "up", and "down" keys). This events occur when a key is pushed down, pushed down and later released, and just released, respectively. This events are, in fact, group of events as each specific key will have its own set of these three events. For example, for the U key of the keyboard there's a "KeyDown U" event, a "KeyPressed U" event and a "KeyUp U" event.

- MouseEvent. Similarly every mouse button press has three events "Down", "Pressed" and "up" events. There are also a pair of "mouseEnter" and "mouseLeave" events that are triggered whenever the mouse pointer enters or leaves the object defining those events.

Co-funded by the Erasmus+ Programme of the European Union

Bear in mind that there is no "Mouse move" event. Therefore if you plan to use the motion of the mouse in your game (for example, an object that follows the mouse), you have to use some other ways to achieve this.

- Create and Destroy events occur at the creation and inizialitacion of new objects (during game) and their destruction.
- Alarm events. These are set to trigger actions that must happen after some time has elapsed.
- Collision events. The game engine is able to solve the speed, direction, position, rotation and shape of every object individually. Besides it can also detect when two objects are touching each other. This is called "collision" because objects are supposedly moving and this situation is expected to impact on the object state, specially on its speed and direction. Everytime a collision happens, The engine fires this collision event.

The Step event deserves a detailed description. This event is particularly powerful, and is very often where the most part of actions occur.

When the game runs, the player might perceive a continuous change in the game, as if changes and evolution happened continuously at any time. But that's just an intended effect, mostly visual. The reality is different because the computers process data and make calculations with digital circuits that run under the periodic signal of a clock. This is, the calculations happen at certain fixed periods. Any program today requires millions of instructions to be processed in short times. Processors are fast and as a result, humans are not able to tell short differences in time when using the computer, hence the continuous flow of change we perceive. But the game engine and game developer need to take processing power into account and control the speed at which the game runs to always produce the same game experience, no matter how fast the computer running the game is. That's not true in other type programs where the faster the better, like for example, databases, web servers or weather simulators.

To allow the developer a precise control of flow of events, the game engine recomputes the evolution of the game exactly sixty times per second (can be changed to thirty). That's roughly 17 milliseconds. For every of those computations, it redraws the screen with the updated information of visible objects. Hence the screen is updated every 17 milliseconds. This is usually expressed as "The game runs at 60 Frames Per Second".

Therefore, the shortest time between game updates is set as 17 milliseconds and the smallest change in game can take no less than those 17 milliseconds to reflect. So if the developer needs to make frequent calculations it doesn't help to do them faster than 60 times per second. Hence, Game Maker has the Step event which is triggered 60 times per second to allow for any pseudo-continuous calculations to be made.

The events by themselves don't change anything, they are just the trigger element of another important part: the actions. To produce an effect, the developer specifies actions

Co-funded by the Erasmus+ Programme of the European Union
The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

and binds the execution of an action to the happening of an event. Every change in the gameplay is produced by the execution of an action which is itself a response to an event.

Summarizing, the goal of the developer or programmer is to choose the relevant events, and write the actions for every one of them. And Game Maker allows for two different ways of specifying actions: using a visual language with icons and symbols than can be dragged, dropped and linked, or by simply writing computer language code.

Co-funded by the Erasmus+ Programme of the European Union
The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.