USING VARIABLES IN GAMEMAKER

A variable is a general programming term for something that can store information. In GameMaker, variables can either store a number or some text. Variables are useful in GameMaker because they allow programmers to store all the unique information about individual objects, such as their position on the screen, or their speed.

Variables are used to store information in the devices memory for later (or instant) use, and they are given a name so that you can refer to them in functions and programs.

A variable is something that we use to store a value for use in one or more operations or functions. Naming things makes life a lot simpler and it also means that should the value of that variable ever change, we don't have to change the number everywhere as the variable name is still the same.

In GameMaker a variable has a name that must:

1. Start with a letter
2. Can contain only letters, numbers, and the underscore symbol '_'
3. Cannot exceed 64 characters.

In many programing languages you need to "declare" a variable before you can use it. This basically means that you tell the computer the name you wish to use so that it can assign a place in memory to store whatever value you decide to hold in that variable. With GameMaker, that is not always necessary as it depends on the scope of the variable. There are four main variable categories when you program with GameMaker and each has its own scope (i.e. its area of operation, or reach).

1. INSTANCE VARIABLES

- These are the most common variables and are defined within an instance.
- These are unique to that instance and can be used in any event and any function within that instance.

2. LOCAL VARIABLES

- These variables must be declared using the **var** function.
- A local variable is only valid for the event or script resource in which it is created.
- So GameMaker will create the variable, use it for the duration of the event and then forget, meaning that if you try to use it later you will get an "unknown variable" error.

3. GLOBAL VARIABLES

- A global variable is one that belongs to the game world itself and not to any one instance.
- It has to be declared as global first, but after that any instance can check or change the variable and the value will always reflect the last operation done on it, no matter what instance or code box has performed the operation.

4. BUILT-IN VARIABLES

- These are special variables that are "built into" the objects and the rooms in the game world and they can be instance only or global in scope (but never local).

INSTANCE VARIABLES

GameMaker uses a number of predefined variables for storing standard information about each object in a game. An **instance variable** is created within an instance of an object and is considered unique to that instance. In other words, many instances of the same object can have the same variable, but each variable can hold a different value as they are unique to each instance. But how is an instance variable created? Well, you create new variables simply by assigning a value to them as shown in the following example:

```
potions = 12;
life = 100;
name = "Jack Black";
strength = 5.5;
armour = -2;
```

As you can see you just have to give the name and then a value (either numeric or a string) to set that variable and have it ready for use within an instance of the object you are coding. These variables can then be used and modified in a number of ways from within the instance, for example this code could be in a collision event and used to take an amount off of the variable **life**:

```
life -= 5 + armour;
```

In the above example, if **life** is at 100 it will now have a value of 97 (100 - (5 + -2) = 97). Now, that's a simple example, and you could replace **armour** for the actual value of -2, but what happens if you use that value in multiple places and then decide to change it? You would have to go through ALL your code and change every -2 to whatever the new value is, which is time consuming and can likely lead to errors. But if you use a variable, all you have to do is reassign it a new value and the code will automatically use that new value from then onwards, making things far more flexible and far easier to fix should there be a problem. It should also be noted that even if a value is not going to change, it is far easier to remember what a variable called **life** means rather than just looking at a number.

LOCAL VARIABLES

A **local variable** is one that we declare first, then use, and then discard.

A local variable is one that we create for a specific event only and then discard when the event has finished. Why would we need them? Well, variables take up space in memory and it may be that we are only going to use them for one operation or function in which case we only need to have it in memory for that short time that it's used. This keeps your code clean and tidy as well as keeping memory space optimised for the things that really need it. To declare a local variable we use the function **var** like this:

```
var i, num, str;
i = 0;
num = 24.5;
```

```
str = "Hi there!";
```

All of the variables created above will be forgotten (i.e. removed from memory) at the end of the event (or script) in which they were created. You must be careful that the name you give declared variables does not coincide with another instance variable within the object running the code, and also make sure that you have no intention of using the value stored in that variable outside of the event from which it was declared. These variables are used a lot in programs, especially in loops for counting how many times something happens, or when using a value several times in one operation that is not going to be repeated again.

GLOBAL VARIABLES

A **global variable** is one that, once declared, belongs to no instance in particular and yet can be accessed by all. Just like local variables, global variables must be declared, but unlike a local variable, a global variable remains in memory until the end of the game. So, you can create a global variable to keep track of (for example) the number of bullets that the player has and then just update this variable at different points in the game. A global variable does not belong to any specific instance and can be accessed, changed and used by all instances at any time, but any change made to the variable are "global", in that all instances using the variable will be affected by the change.

As with local variables you have to take care not to name your global variables the same as any instance variables as that will cause you problems and make bugs creep into your games. We can actually call global variables in another way to help with this problem, and that is by using the word **global** and a period (".") before the variable.

In the following example, we have declared a global variable called "food":

```
global.food;
```

GameMaker has a collection of "built in" global variables too, so you should be aware of them as you may name one of your instance variables the same or wish to have your own global variable with the same name and wonder why you are getting errors. They are easy to spot, however, as they are shown in a different colour in the code editor and also come up in the auto-complete bar at the bottom. The majority of built in global variables are very specific and will only be used on rare occasions, but there are three in particular which are very useful:

- **Score** – used for storing the game score.
- **Health** – used for storing the player health.
- **Lives** – used for storing the player lives.

BUILT-IN VARIABLES

GameMaker uses a number of predefined variables for storing standard information about each instance in the game. There are way too many to name them all here. But you can find a complete list of GameMaker's built-in variables by clicking on the following link:

http://docs.yoyogames.com/source/dadiospice/002_reference/objects%20and%20instances/instances/instance%20properties/index.html