USING PARENT OBJECTS IN GAMEMAKER

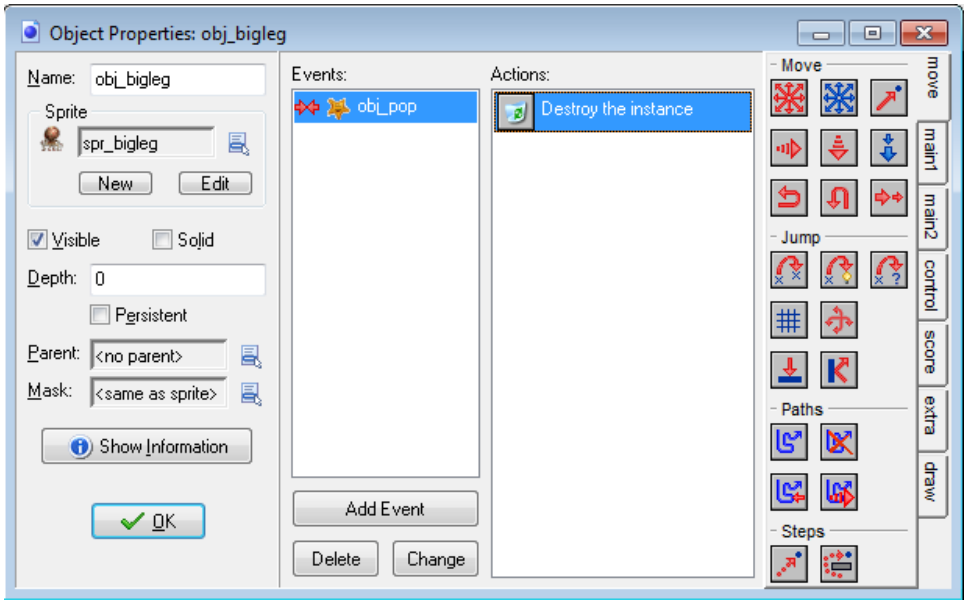
One of the most powerful options within the object properties is the ability to assign a **parent**. Every object in a game can have a parent object, but what does this mean? Well, when an object has a parent, it can share code, actions and events with that parent. This is called **inheritance** and an object that has a parent is called a **child**. But that's not all! You can also do checks and run code on parent objects which automatically include the child objects too which saves a lot of time and energy. Another way to look at a parent object is as a way to "group" objects together under the same umbrella and have them share certain things without losing their own identity. Now, this may seem a bit complicated to understand so let's give some examples:

Say you have a player object and four different enemy objects. Now, you want the player to die if he touches any of these four objects and this would normally entail four collision events with four sets of actions or code, one for each of the enemy objects. BUT if we make a parent object for all the enemies, then we can create one collision event with the parent object only and it will trigger no matter which one of the four "child" enemy objects touch the player.

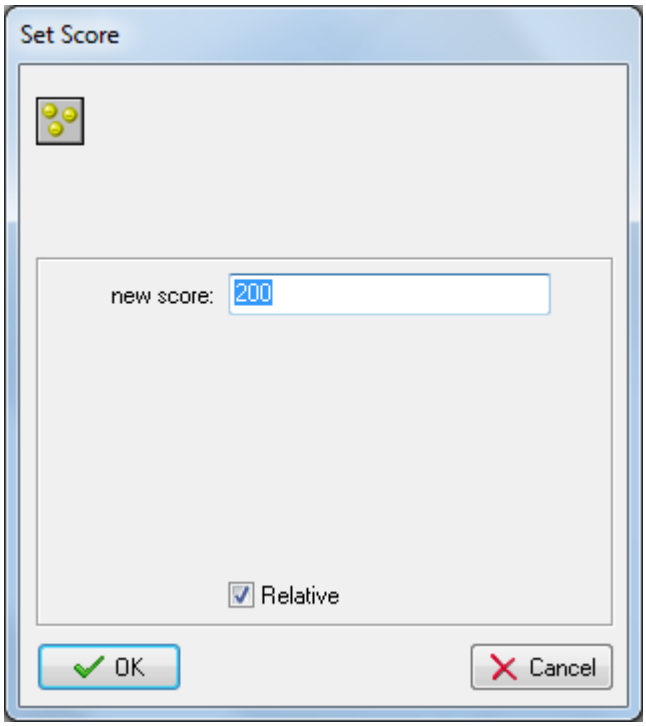
Let's create two enemies for a game we'll call Super Rainbow Reef. The resources for the game have already been provided and can be found on the shared directory.

CREATING A BIGLEG OBJECT

1. Create an object called **obj_bigleg** and give it the sprite **spr_bigleg**.
2. Add a **Collision** event with **obj_pop** and include a **Destroy Instance** action.

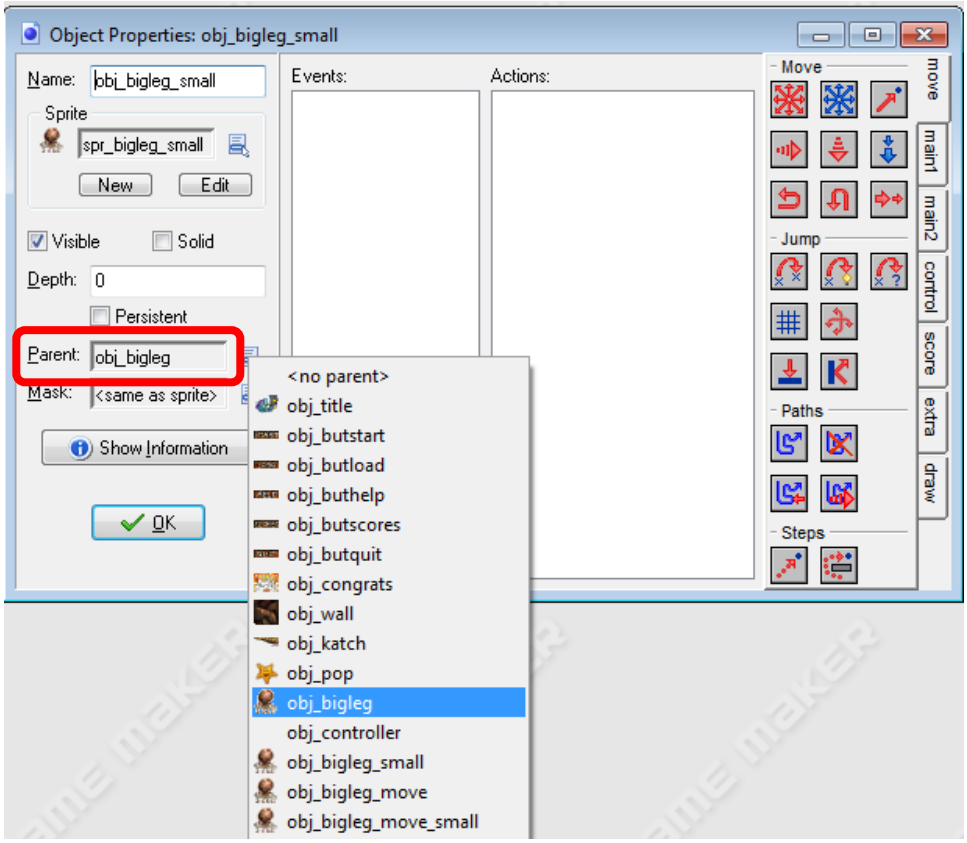


3. Include a **Set Score** action, with **New Score** set to 200 and **Relative** enabled.



CREATING A SMALL BIGLEG OBJECT

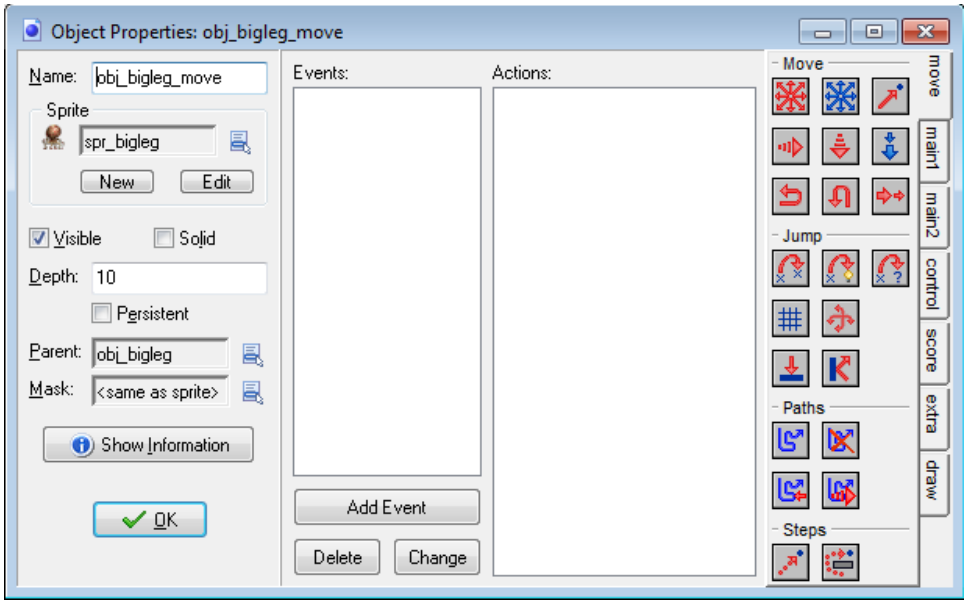
1. Create an object called **obj_bigleg_small** and give it the sprite **spr_bigleg_small**.
2. Click the menu icon next to the **Parent** field and select **obj_bigleg** as the parent.



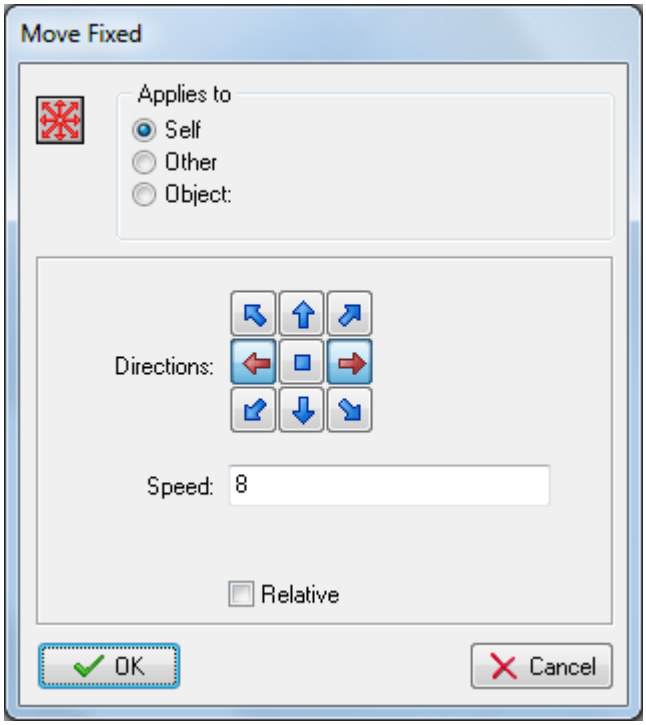
We now have a small Bigleg object that does everything that the normal Bigleg object does. This is because the small Bigleg inherits the behavior (all the events and actions) of the normal Bigleg when we make the normal Bigleg its parent. This means it reacts in the same way to collisions with Pop and will automatically know to increase the player's score by 200 and destroy itself. There's no need to add these events and actions again since they are automatically inherited from the parent. Also, as a child of the Bigleg, the small Bigleg is now considered to be a Bigleg too.

CREATING A MOVING BIGLEG OBJECT

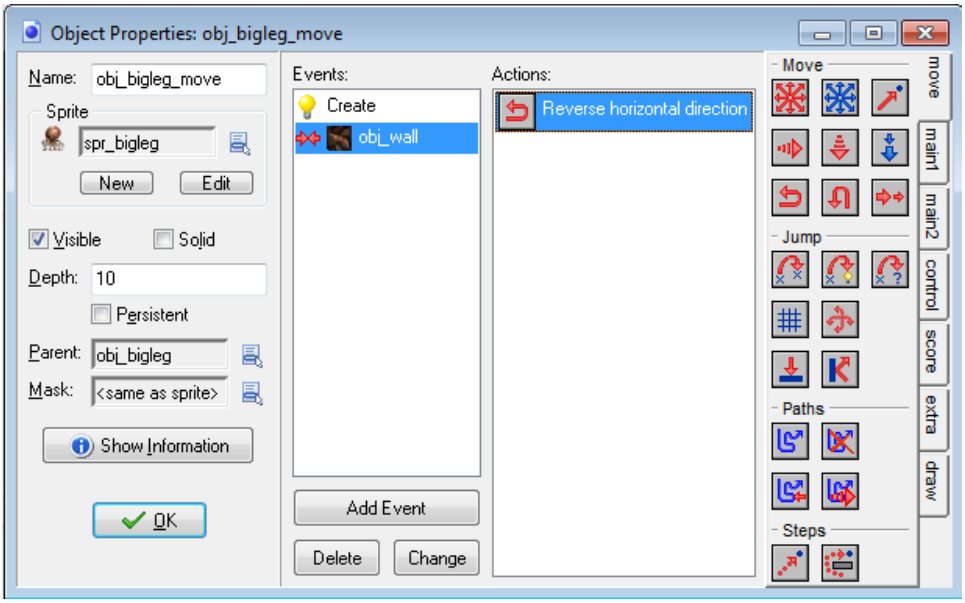
1. Create an object called **obj_bigleg_move** and give it the sprite **spr_bigleg**.
2. Click the menu icon next to **Parent** and select **obj_bigleg** as the parent. Also set the **Depth** field to 10 so that it appears behind other objects.



3. Add a **Create** event and include the **Move Fixed** action in it. Select both the left and right directions and set **Speed** to 8.



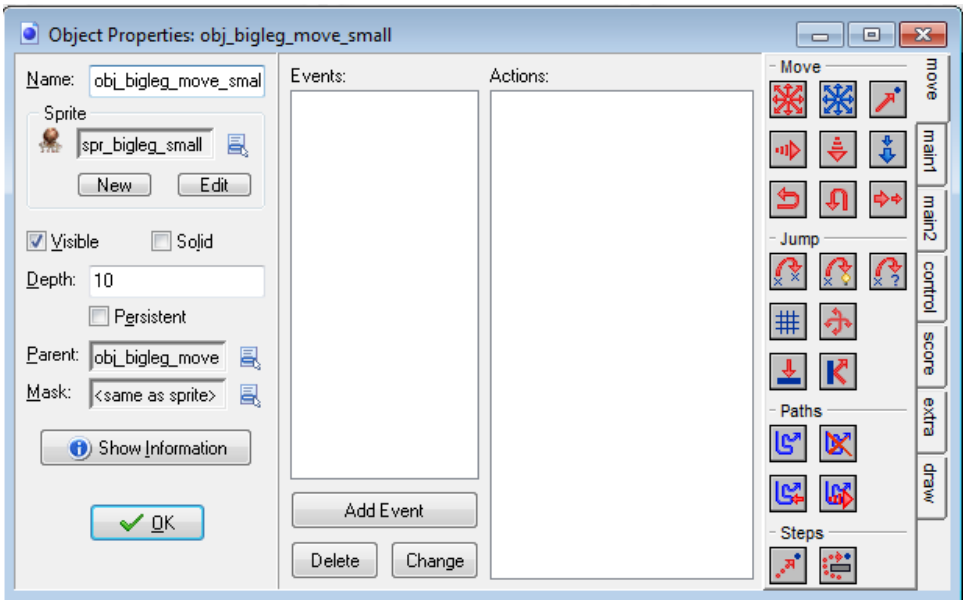
4. Add a **Collision** event with the wall object and include the **Reverse Horizontal** action in it.



This moving Bigleg automatically inherits the behavior of the normal Bigleg (to destroy itself when Pop collides with it) and adds to that some extra behavior to make it move back and forth. In the same way, we can quickly add a small moving target.

CREATING THE SMALL MOVING BIGLEG OBJECT

1. Create an object called **obj_bigleg_move_small** and give it the sprite **spr_bigleg_small**.
2. Click the menu icon next to **Parent** and select **obj_bigleg_move** as the parent. Also set the **Depth** field to 10 so that it appears behind other objects.



This time we're using the moving Bigleg as the parent, which in turn has the normal Bigleg as a parent. This means it inherits the behavior of both the moving Bigleg and the normal Bigleg.

THE POWER OF USING PARENT OBJECTS

As you have seen, parents are extremely powerful and can save you a lot of time. The Biglegs are quite simple, but parents can save you hours of repeated work for objects with many events and actions. You could just duplicate objects to save time, but any changes you want to make afterward have to be made to both the original and each of the copies you made. However, when you change a parent, the changes automatically apply to the children of that parent too – a very useful feature.

So here is a summary of rules that you need to understand moving forward when working with parent objects:

INHERITING EVENTS

A child inherits all the events (and actions) of its parent. A child can have its own events as well, but these only apply to the child and not the parent. When both the child and parent have the same event with different actions, then the actions of the child are used for the child, and the actions of the parent are used for the parent.

ACTIONS ON OBJECTS

When an action refers to a parent object, this includes instances of the child object as well. However, when an action refers to the child object, it does not include instances of the parent object.

COLLISIONS WITH OBJECTS

A collision event with a parent object also applies to collisions with children of that object. However a collision event with a child object does not apply to collisions with parents of that object.

PARENTING OBJECTS

Parents can have parents, which can have parents, etc. However, you must not create cycles of parents, so if P is the parent of C, then C cannot also be the parent of P.