



Game Baker

Open Source, Cross Platform Game Editor

Written for Release 0.1.3

<http://game-baker.googlecode.com>

Tutorial : Using the Parent Property

This tutorial will introduce the concept of a parent workstate in Game-Baker. This is a very powerful concept, and is especially useful for instructors and teachers. By creating custom workstates as part of a lesson plan you can allow your students to create more advanced games much more easily, while hiding the advanced scripting in a different object, and simply instructing them to set the parent property to your pre-written workstate.

We are going to create multiple objects, each of which will start in a different position on the screen, but will move in the same way. By the end of this tutorial, you should know how to:

- create a parent workstate
- Create multiple child workstates, each of which inherit their behavior from the parent.

Note: The Buttons on your version of Game Baker may look a little different, depending on which operating system you are using, the release of Game Baker you are using, and your machines display settings.

Step 1: Open Game Baker

Open Game-Baker by executing the `gamebaker.py` file in the `/trunk` directory that you have installed Game-Baker to. If you have installed via the Deb package, you can access Game-Baker through the main menu.

Step 2: Create a New Game

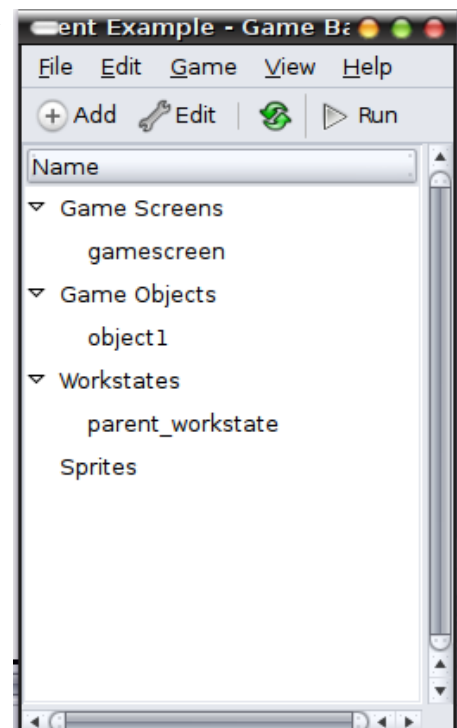
Press `Ctrl-N` or go to `File > New`

Now Create a suitable title for your game, and choose a good name for the first gamescreen. For this tutorial, I have named the game `Parent Example`, and the gamescreen `gamescreen`.

Step 3: Add a new workstate, and a new object

By double clicking on the objects, you can name them whatever you desire, for this tutorial we will name them `parent_workstate` and `object1`.

The list of objects on the main window should now look something like the image on the right.



Step 4: Create a event handler for the ITERATION event.

Select `parent_workstate` , and click `Add Event > Iteration` to add an iteration event. Now select this event using the box at the top.

We are going to enter the following event code:

```
self.move_velocity()
```

, which will move whichever object has this workstate according to the properties `self.vx` and `self.vy` (x and y velocity respectively).

Make sure you click `Save Event`

Step 5: Create an event handler for the OFFSCREEN event

Create another event, but this time choose `OFFSCREEN` . This event is called whenever an object is not on the screen any more. The code we will enter this time is:

```
if self.x < 0 or self.x > gamescreen.width:  
    self.vx = - self.vx  
if self.y < 0 or self.y > gamescreen.height:  
    self.vy = - self.vy
```

, which will make the object appear to `bounce` off the sides of the screen.

Make sure you click `Save Event`

Step 6: Test Your Workstate

In order to test your workstate, we will need to add a couple of lines to the `INIT` event of the workstate. Switch to this event (which is automatically created), and enter the lines

```
self.x = 0  
self.y = 0  
self.vx = 5  
self.vy = 5
```

, which will position our test object, and set up the initial velocities.

Save this event, and click `Ok` to close the workstate.

Now double click on `object1` that we created earlier, and set the initial workstate to be `parent_workstate` . Click `Ok` to shut the window.

The final thing to do it to add this object to the gamescreen. Double click on `gamescreen` that we created earlier, select `object1` and click `Ok` .

Now click `Run` - and you should see the object you created bouncing off the walls.

Step 7: Create Two more objects.

We can now add some child workstates. Add two more workstates. This time, the only events we need to write for the workstates are `INIT`. For one workstate write the following `INIT` event:

```
self.x = 20
self.y = 30
self.vx = 4
self.vy = 4
```

, and for the second workstate add the following:

```
self.x = 25
self.y = 5
self.vx = 2
self.vy = -4
```

This will position the two objects at different places, and start them off with different speeds.

Now create two new game objects, and set the initial workstates for them to be the ones that we have just created.

Add these objects to the game screen, and click `Run`.

You should see that the first object we created is moving around and bouncing, but the new two are not moving yet. Time to change that.

Step 5: Set the `Parent` property

For the two new workstates you made, open them up and in the box marked `Parent` enter the name `parent_workstate` (the name of the workstate we created earlier). Click `Ok` to close the windows.

Now click `Run` ... all three objects are moving around the screen, bouncing off the sides.

What has happened is that Game Baker has noticed that the new workstates don't have the `OFFSCREEN` event, and so that event gets taken from the parent... clever, isn't it.

Summary:

You can use this for several things but if you are instructing a class this is especially useful you can add as much complicated code as you want to a parent workstate, and to all your class have to do is to set the `Parent` property of their workstates to be the name of your parent!



If you feel that Game-Baker needs a tutorial on a specific topic, just post a request in the forums or give one of us an email. You can also place it as a comment on the bug report for tutorials [here](#).

Remember, you are always welcome to help out.