CONTAINERS

View online: https://www.construct.net/en/make-games/manuals/construct-3/project-primitives/objects/containers

Containers are an advanced feature to help build *composite objects* - that is, elements of your game made from multiple objects. For example, a tank in a strategy game might be composed of two parts: a sprite for the tank base, and a different sprite for the tank turret. This allows them to rotate independently. Adding them both to a container then allows events to treat both objects as if they were one, because they are always picked together.

> ***Hierarchies** are another feature that help with building composite objects (see Setting up a hierarchy in the Layout View). Containers generally apply to picking groups of objects in event sheets, whereas hierarchies generally apply to making objects move and rotate together. Both features can be used together as well.*

It is essential to be familiar with how events work in order to understand how containers work.

Creating a container

To add an object to a container, select one of the objects you want in the container and click the **Create** link in its properties (which appears under the *Container* category next to the label *No container*). A dialog opens allowing you to choose the object to add to the container.

Further objects can be added to a container by clicking the *Add object* link in the *Container* category again. Objects can be removed by clicking the *Remove* link.

What containers do

Placing objects in a container has the following effects:

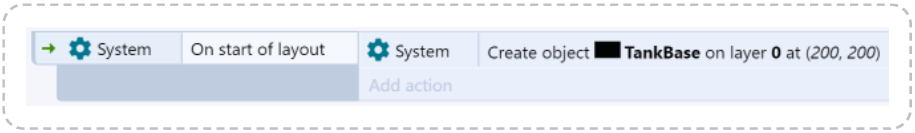
1. If one object in a container is created, every other object in its container is also automatically created.
2. If one object in a container is destroyed, every other associated object in its container is also destroyed.
3. If a condition picks one object in a container, every other associated object in its container is also picked.

The first two points basically guarantee that there is the same number of instances for all the objects in a container. In other words, containers are created and destroyed as a whole. Using the tank base and turret container example, it is impossible to create a tank base without also automatically getting a new turret for it as well.

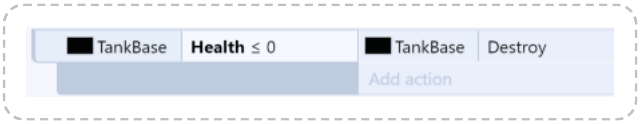
The third point is the main purpose of containers. Containers are also *picked* in events as a whole. This makes events treat containers as if they were one object. For example, if a condition picks a tank base instance, it also automatically picks the base's associated turret.

Examples

In the following events, assume both *TankBase* and *TankTurret* are in a container.



In the above event, a *TankTurret* instance is also created since it is in a container with *TankBase*. It might also be useful to add an action to set the position of the *TankTurret* to make it appear on top of the base.



In this event, the associated *TankTurret* instance is also destroyed since it is in a container with *TankBase*.

In this event, when a bullet hits the tank base, only its associated turret flashes. If the objects were not in a container, *all* the turrets in the game would flash, as per the rules of how events work (since no turret was referenced in the conditions, the action applies to all of them). However, since the objects are in a container, when the *TankBase* that was hit by a bullet is picked, its associated turret is also picked. This makes the event work as intended, and the event treats both objects as if they were one. This is the crux of containers, and for some uses like strategy games, there will be a large number of events taking advantage of this type of picking to ensure objects work as units and don't accidentally affect other instances.

Placing container objects in a layout

In the Layout View, it's possible to create an instance in a container by itself. This appears to break rules 1 and 2 under *What containers do*, since objects in a container must always create and destroy together. However, the editor does not enforce this. Instead, any missing objects are created automatically when the layout starts. It is a good idea to make sure you create enough objects anyway so you can edit the object's position, instance variables, and other properties from the Layout View.

Using containers in the Layout View

In the Layout View, when you highlight an instance in a container, the other instances in the same container highlight in yellow to help you identify which instances are grouped together.

You can also change the *Select mode* property to one of the following:

- **Normal:** instances in the container select individually like normal.
- **All:** instances in the container all select together whenever you select one of them. This is like always multi-selecting every instance in the container whenever you click one of them.
- **Wrap:** also selects all instances in the container, and then also wraps the selection so they stretch and rotate as one. This means you can treat containers as if they are one object. For more information see *Selection wrapping* in the Layout View.

You can circumvent the selection mode by holding `Alt` and selecting an instance. This will let you select just that instance even when *Select mode* is *All* or *Wrap*. This helps you change the container as well, such as to add or remove another object type to the container.

Data storage objects in a container

It's possible to add data storage objects like Array and Dictionary to a container with another object. Despite the fact these objects are invisible, a separate instance of the object is still created for each container. This allows you to have a dedicated Array or Dictionary for each instance of an object. This can be very useful as an advanced substitute for instance variables, such as if a very large number of variables is necessary, or if variables need to be dynamically added and removed.