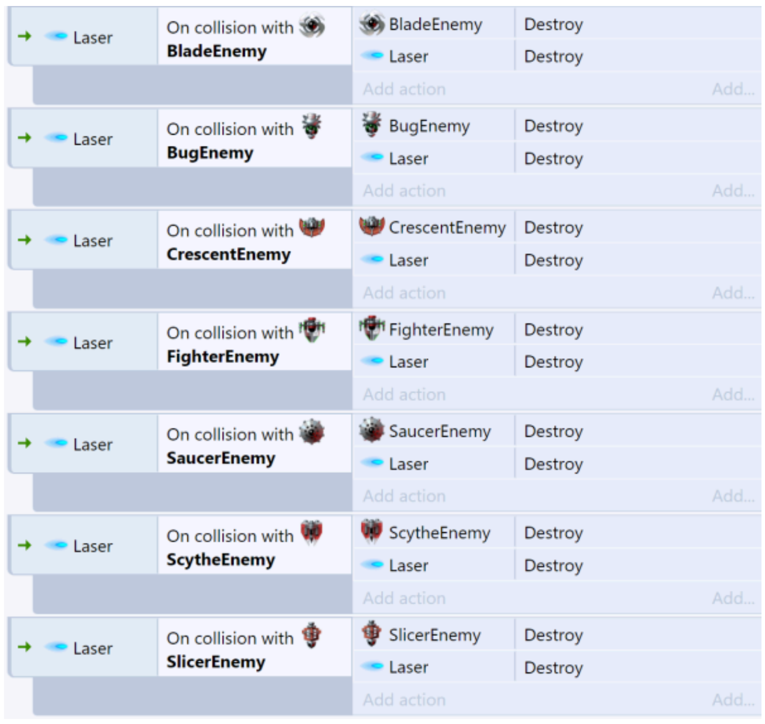
FAMILIES

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

Paid plans only In Construct, **Families** are groups of object types. All the object types in a family must be from the same plugin, e.g. all Sprite objects (and not a mix of Sprite and Tiled Background objects, for example).

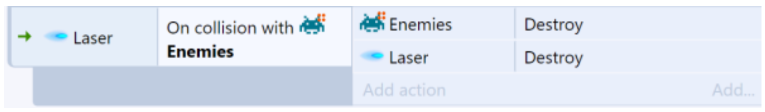
Families can help you avoid repeating events. For example, instead of having the same events for the *Enemy1*, *Enemy2* and *Enemy3* objects, you can add them all to an *Enemies* family and make the events once for the family. Then, the events automatically apply to all the object types in the family.

The Families example in Construct demonstrates the advantage of this. There are seven kinds of enemy, and they all need to be destroyed when the laser hits them. Without families, seven separate events are necessary, as shown below:



Laser	On collision with **BladeEnemy**	BladeEnemy	Destroy
		Laser	Destroy
		Add action	Add...
Laser	On collision with **BugEnemy**	BugEnemy	Destroy
		Laser	Destroy
		Add action	Add...
Laser	On collision with **CrescentEnemy**	CrescentEnemy	Destroy
		Laser	Destroy
		Add action	Add...
Laser	On collision with **FighterEnemy**	FighterEnemy	Destroy
		Laser	Destroy
		Add action	Add...
Laser	On collision with **SaucerEnemy**	SaucerEnemy	Destroy
		Laser	Destroy
		Add action	Add...
Laser	On collision with **ScytheEnemy**	ScytheEnemy	Destroy
		Laser	Destroy
		Add action	Add...
Laser	On collision with **SlicerEnemy**	SlicerEnemy	Destroy
		Laser	Destroy
		Add action	Add...

Using families all seven events can be replaced with a single event:

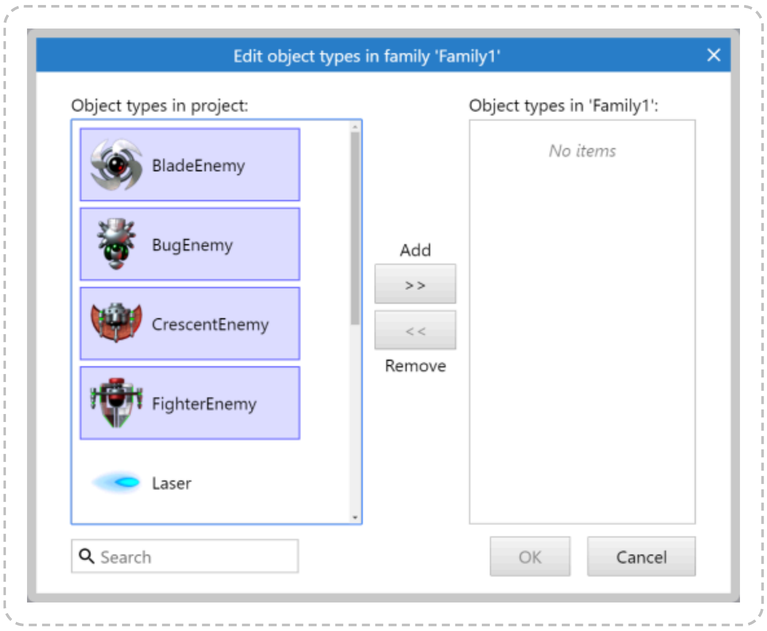


Laser	On collision with **Enemies**	Enemies	Destroy
		Laser	Destroy
		Add action	Add...

This makes it far easier to create and maintain projects with lots of objects that need to work in similar ways.

How to create a family

Right-click the *Families* folder in the Project Bar and select **Add family**. The *Edit Family* dialog appears.



Objects on the left are the objects in the project that can be added to the family. Objects on the right are the objects already in the family. **Double-click** an object to transfer it to the other side. You can select multiple objects by holding `Control` and clicking several objects, then clicking one of the buttons in the middle to transfer them.

When done, click **OK** and the family will appear in the Project Bar. It can be expanded to show all the objects in the family as well. The family, and the objects in the family, can be edited by **right-clicking** them and choosing options from the menu, like *Remove from family* or *Edit family*.

Objects can be added to multiple families. All events for the object's families will apply to the object.

Family instance variables

Instance variables can also be added to a whole family by **right-clicking** the family name in the Project Bar and selecting Family instance variables.

If you add an instance variable to a family, *all* the object types in the family inherit the instance variable. For example, adding the instance variable *health* to the family *Enemies* in the above example will mean *BladeEnemy*, *BugEnemy*, *CrescentEnemy*, *FighterEnemy*, *SaucerEnemy*, *ScytheEnemy* and *SlicerEnemy* all gain a *health* instance variable. It will also appear in the editor alongside each object's own instance variables. However in the Event Sheet View the family will only show its own instance variables (those added directly to the family). This means any instance variables you want to be available to the family's events must be added to the family, and not to the objects in the family.

If an object type belongs to multiple families, it inherits every family's instance variables.

Family behaviors

Behaviors can also be added to a whole family by **right-clicking** the family name in the Project Bar and selecting Family behaviors.

As with family instance variables, if you add a behavior to a family, *all* the object types in the family inherit the behavior. The behavior will appear in the events for all the objects in the family *and* the family itself. For example adding the *Bullet* behavior to a family called *Bullets* means the bullet's *Set speed* action is available to every object in the family, as well as the family itself.

If an object type belongs to multiple families, it inherits every family's behaviors.

Family effects

Effects can also be added to a whole family by **right-clicking** the family name in the Project Bar and selecting Family effects.

As with family instance variables and behaviors, if you add an effect to a family, *all* the object types in the family inherit the effect. This can be useful for quickly applying effects to a number of different object types.

If an object type belongs to multiple families, it inherits every family's effects.

Picking families in events

Families pick instances in the event sheet independently of the object types in the family. For example, consider *Family1* consisting of *SpriteA* and *SpriteB*. Conditions for *Family1* will never affect which *SpriteA* and *SpriteB* instances are picked. It will only affect which instances are affected when running an action for *Family1*. Likewise, conditions picking *SpriteA* and *SpriteB* instances will never affect which instances are picked in *Family1*. In other words, in the event sheet families are treated like an entirely separate object type, which just happens to have instances from other object types. This can be taken advantage of if you need a single event to pick two separate lists of instances from the same object type.

Upgrading ordinary objects to families

If you make a lot of events forgetting to use a family and want to replace them, it's possible to use the *Replace Object* feature to save you re-doing every event. The process is described in the tutorial How to upgrade an object to a family.

Summary

Families are a very powerful feature which are essential to help keep large projects simple. Instance variables and behaviors added to families are inherited by every object in the family, which allows for sophisticated logic to be easily applied to many object types at once.