Prof. Monster's

BGE Guide to

BGE Libraries

By Monster
Version 1.1
23 Jun 2014

When it comes to larger Game Projects you need a way to organize it's content. A very nice method is to use other blend files as libraries.

This guide will explain what libraries exist and how to use them properly.

I hope you find it useful

Monster

Purpose

23 Jun 2014

- added this History page
- added API References page
- updated Instance Properties page

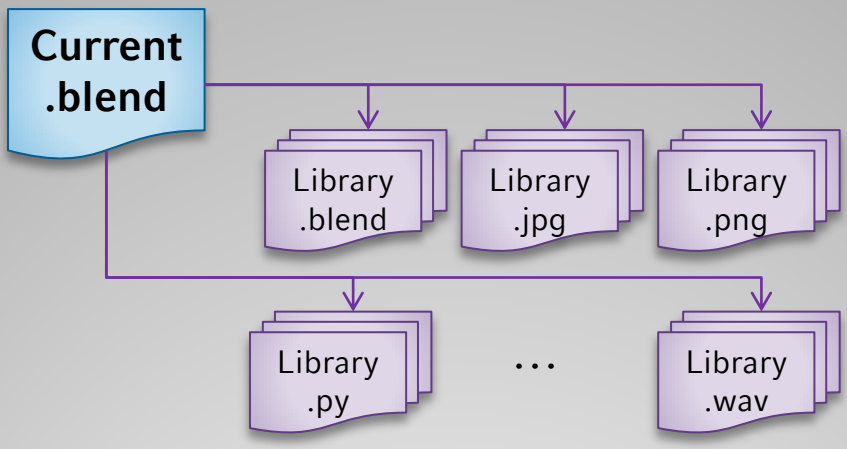
02 Jan 2012

- initial Version

History

There is no official term **library** for the Blender Game Engine. Let us define it:

- A BGE Library is a data block that reside in a different file than the current executed blend file (in blender referred as external data).



Libraries can be any type of file that is supported by Blender.

Within blender linked data blocks are marked with a link symbol.

Linked libraries can be referenced relative or absolute. It is recommended to ensure all paths are relative before publishing the game. Otherwise the libraries might be located at a different location.

What is a library

Linked data will be updated automatically after reloading the linking file. Within the BGE the data will be updated after loading the game.

We will focus on blend files with game objects as content.

While game objects can be linked directly from a blend file, this method is not very flexible.

Linked game objects can only be changed (during design time) within the file they reside in. That makes it hard to place them at the proper location within a level file. Also scale and rotation can not be changed within the file they are linked to.

A better method is to assign the game object to a group and link the whole group. This group can be instantiated multiple times and placed at any location without effecting the library.

Groups

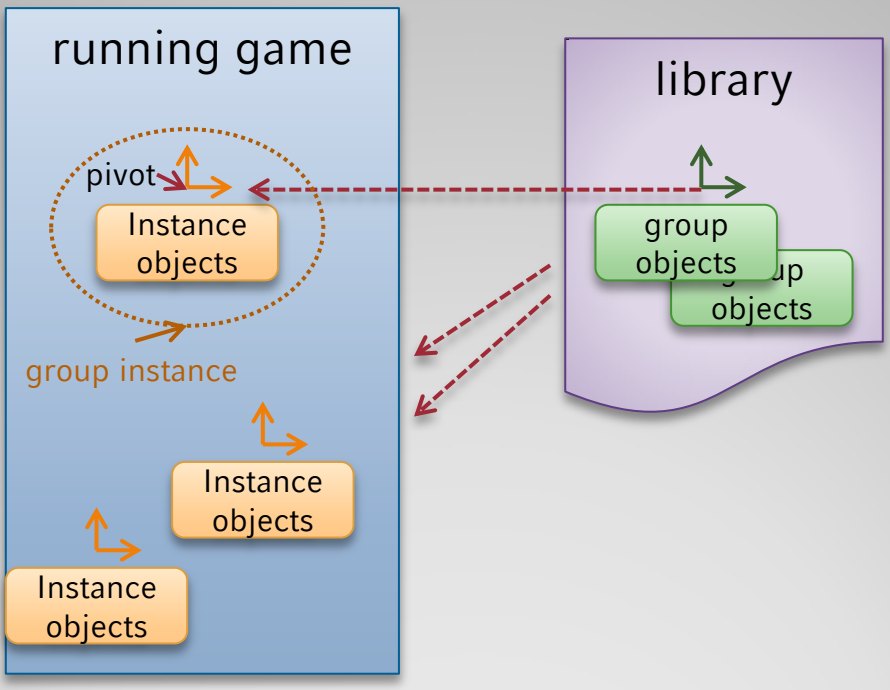
Now we have game objects with different purpose. Let's give them names:

group object a Blender object that is assigned to a group - it acts as template for instance objects.

instance object an object that is a copy of a group object created during instantiation.

pivot the instantiating object - it creates the instance objects via doupli group (usually an empty).

group instance a pivot and all instance objects.



Groups

Groups are perfect for building Libraries:

- Groups provide a single access point to the objects of the groups
- Groups hide the implementation details.
- Instances can be part of another group
- There can be any number of instances of a group.
- Instance objects know they are part of an instance and the other members of it while in the BGE.
- Instance objects know the pivot object that added them.
- Pivot objects know the instance objects they added.

Remarks:

- Groups (as an entity) do not exist when running a game. The instance objects are added to the BGE while converting the Blender scene into a BGE scene (when loading the scene).
- Groups are not accessible when running the BGE (but the instance objects are).
- Instance objects do not know the group or group name while in the BGE.
- Instance objects are not parented to the pivot (usually you do not need that). If you move the pivot when running the BGE the instance objects stay where they are. This is different to the behavior within Blender
- Children of instance objects do not know they are instance objects too nor are the known to the pivot.
- Instance objects do not have unique names.

Instance properties

Since Blender 2.64a the BGE API supports group instance. This allows us to be much more flexible with them.

There are two attributes to access an instance:

- **groupObject** refers to its pivot object
- **groupMembers** is a list with references to the instance member objects of a pivot object .

Be aware pivots can be members of another instance.

```
owner = getCurrentController().owner
ownersInstantiatedObjects = owner.groupMembers
ownersPivot = owner.groupObject
ownerAndBrothers = owner.groupObject.groupMembers
```

This allows us some nice options to create custom logic bricks:

```
from bge.logic import getCurrentController

def parentOwnerToPivot():
    owner = getCurrentController().owner
    ownersPivot = owner.groupObject
    owner.setParent(ownersPivot)

def parentPivotToOwner():
    owner = getCurrentController().owner
    ownersPivot = owner.groupObject
    ownersPivot.setParent(owner)
```

(Both functions are meant to be set up within the group rather than at the instance)

Attention: *Children and Grand-children of an instance member are not listed in groupMembers nor do they have a groupObject set regardless if they are part of the group or not.*

API

You read how the Game Loop runs in the BGE. This should provide you with an idea what runs when in your Game.

I hope it helps you to create a fancy game without discovering to much obstacles on your way.

The road to success is long and I hope this little information can point you into the right direction.

You can find me under www.blenderartists.org as Monster

Good luck

Monster

Thank you