Better Scene Modding 101

The easiest way to start modding is to use existing scene asset for monkey plugin. BetterScene can detect monkey presence and expose assets it can load as scene packages. This tutorial is based on TheFlatV2 Monkey asset by Manniakk.

BetterScene internally called CustomScene, CS for short. CustomScene and BetterScene are interchangeable names.

Creating new modification package using monkey asset

Create new directory in Packages. Directory name is not important for loader, but it is good idea to use mod id as name. Recommended mod id should contain author and actual id: <author>.<id>.

Next, you need to create package manifest manifest.json. Minimal manifest may look like this:

```
{
	"type": "story",
	"id": "<your mod id>",
	"name": "<your mod display name>",
	“author”: “<author’s name>”,
	"version": "1.0.0",
	"require":{
		"monkey.the_flat_v2_evening":"0.0.0",
		"basic_poses": "1.0"
	},
	"plugins":{
		"com.thora.monkey":"1.8.7"
	},
	"mainScene": "myscene"
}
```

Package *type* can have following values:

- asset – generic asset package that can be referenced by other packages. Use-cases: graphics assets, poses, etc
- extension – this type of package allows to expand contents of specific packages. Use-cases: additional poses for postures provided by asset package; replacing content
- story – package that can be selected from menu and started

Plugins section is used to declare BepInEx dependencies. In this case package require monkey.dll, to load monkey assets.

Require section allows to declare package dependencies. Dependencies are mandatory. In this example modification requires to load package with basic poses and actual monkey asset.

BetterScene uses the following pattern for monkey mod id: monkey.<assetfilename>. For example: “theturtle.unity3d” file will be exposed as “monkey.theturtle”.

mainScene parameter controls scene id to load when story starts.

Finally, you need to define scene file myscene.scene with following content:

```
{
	"include_before":[
		"monkey"
	],
	"sequence":[
		{ "type": "destroy_go", "name":"Eviroment (2)/Staticos/Officina1 Variante/WallLogic/Statics/TranslacionTargets/"},
		{ "type": "destroy_go", "name":"Eviroment (2)/Staticos/Officina1 Variante/DeskLogic/Statics/TranslacionTargets/"}
	],

}
```

Scene file is a sequence of operations required to load final scene.

- *include_before* allows to load additional scene files before main sequence. In this case it is monkey.scene (generated for monkey asset and provided by monkey.* package).
- *sequence* parameter contains list of scene loading operators. In this case it purges vanilla GoTo targets because they have no sense for custom scene.

During scene loading BetterScene automatically cleans up references to missing GoTos to prevent crashing.

Now story can be launched from Custom Story Menu.

Troubleshooting:

main log is BepinEx/LogOutput.log

- Package is not listed – see logs for json related errors
- Loaded office instead of custom scene – see logs for scene loading errors
  - wrong mainScene id
  - json errors in scene file
  - some other errors
- Package listed but can not be launched – see errors info in package description window
  - missing dependency
  - wrong monkey package id: see logs for available package ids

Setting up scene

BetterScene defines 4 object types to describe scene:

- POI – point of interest. A place character can go to. Examples: Desk, Desk Side, Wall
- Posture – primary character pose. Other poses are defined in relation to character's posture. Examples: Sit, Stand, Bend, Lie.
- Pose Clips – exact poses available for character at specific posture
- POI Posture – exact posture implementation in scene. Examples: to *Sit* on *Table* at *Kitchen*. Where Sit is posture, Kitchen – POI, and Table – is exact point this pose available at.

Example situation:

Waving hand while sitting on kitchen table.

Kitchen is POI
Sitting on table is posture
sitting on kitchen table is POI posture
Waving hand while sitting on table is Pose Clip

Implementation details:

- POI – defined as position and rotation in world space
- Posture – bone rotations, hips offset and root offset. Used as template only
- POI Posture – bone rotations, hips offset and root offset. This disposition is used as default idle pose when no Idle clip provided
- Pose Clip – collection of bone rotations and hips offsets. Root offset is applied from POI posture data

Scene Tool Overview

Every scene object is altered from Custom Pose view. BetterScene uses vanilla pose editor that has own specifics. Because of that it is recommended to use “willing” character to edit scene, otherwise there may be problems with bone movement.

Switch character to Custom Pose mode. You will see Additional window on the left side called “Scene Tool”. This window is the main toolbar for every BetterScene related thing.

Bones tool is posing helper tool.

When you select bone, you can see it’s name, coordinates, and use some buttons. Brief description of this buttons:

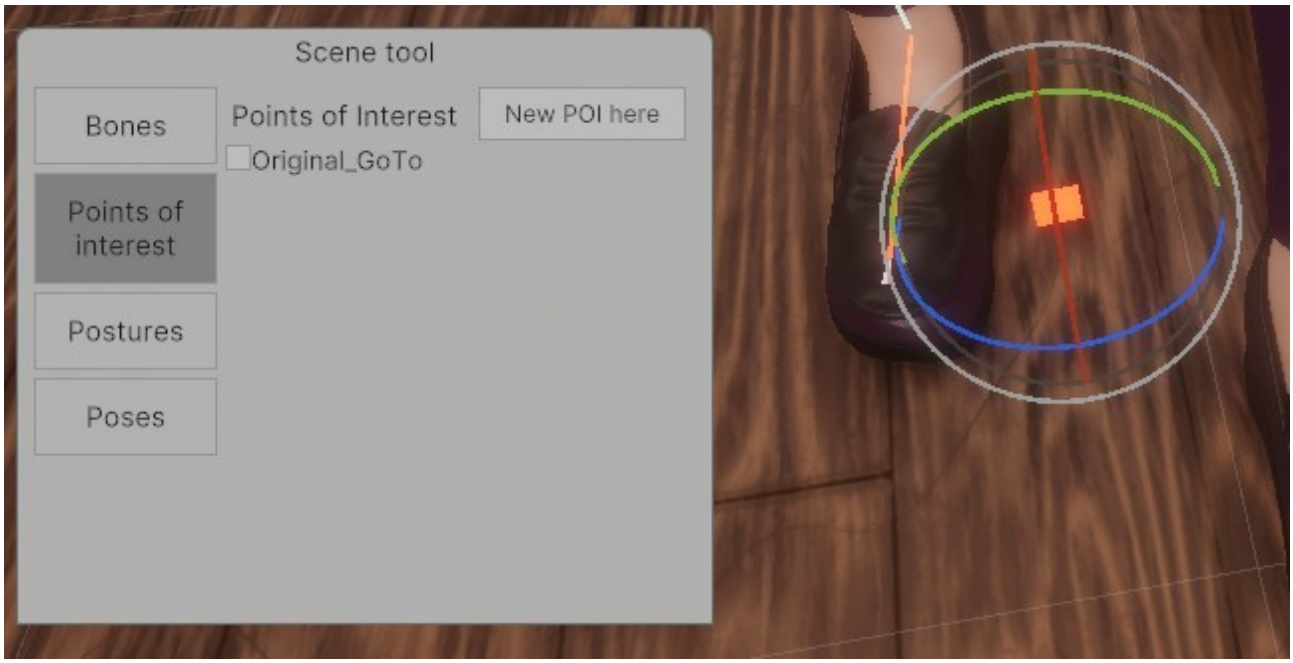
- L<>R – switch between left and right bone
- ASC – switch to ascending bone, aka parent bone
- DSC – switch to descending bone, aka child bone (single child bone only)
- FWD/BWD – add or subtract euler rotation from Rel.Rot. Field
- SYN – synchronize opposite bone’s rotation, aka synchronize direction
- MIR – mirror rotation. Same as SYN but with mirror symmetry.
- TSYN/TMIR – Tree[SYN/MIR]. Applied to selected bone and it’s subtree, aka sync limb.
- TCPY/TPST – copy/paste subtree rotations. Useful when creating multipose clips to transfer limb orientation.
- Hips locked/unlocked – allows to move(but not rotate) hip bone without dragging all hierarchy.

Other tool windows share same structure: they have “New” button to create new entity and list of objects. Each row contains object id and control button. That button can be used to perform object specific actions like “apply”, “edit” or “remove”.

Creating POI

To start, create new POI at Sofa. Switch character to custom pose mode.

Drag root bone to desired position and choose proper rotation. Orientation is important, it will be used as main frontal orientation(turn around action). As rule of thumb, by default character should be facing player in front of object of interest.



Then press [new POI here] and type POI ID. In this case Sofa.

> ID must be Latin alphanumeric, without spaces and special characters. There are no validations. Just don't.
>
> *POI Id is not a name*. It is identifier to reference this POI

Next, click on button near created POI and select "Edit descriptor".

Type "Sofa" in Display Name and press Save.

> Descriptors – are additional data associated with object. Descriptors are designed to be human readable json files.
>
> When Display Name is not specified, CS will use object ID as its name. It may work in case of Sofa, but it won't work in other cases, like Original_GoTo on picture above. It is recommended to assign proper Display Name for every entity.

Additionally, POI can have "parent POI":
When Kitchen_Table has Kitchen as parent POI, you can go to Kitchen_Table only from kitchen. This allows to reduce go to list.

This use case scenario will be covered later.

Creating POI Posture

BetterScene is bundled with basic poses package. It contains some poses from original game, and postures to be used with BetterScene.

Postures aren’t poses. Each posture is made with some assumptions in mind. That assumptions are related to poses posture hosts. From this perspective posture can be seen as “animation host” that contains poses available at some specific object of interest. Main role of postures is “create once apply everywhere”. This way you can create postures for common objects of interest, and just implement them at various POIs by editing root offset.

Let’s teach character to sit on sofa. Basic poses contains 2 to postures related to sitting. Both Sit posture and Climb implement different kinds on sitting pose.

Sit posture describes sitting on a small horizontal surface with very little space. Best example is chair. Character can sit, but cannot do anything else.

Climb posture is sitting on some relatively large horizontal surface. Example is bed, table, some couches. This posture suits best for scene’s sofa.

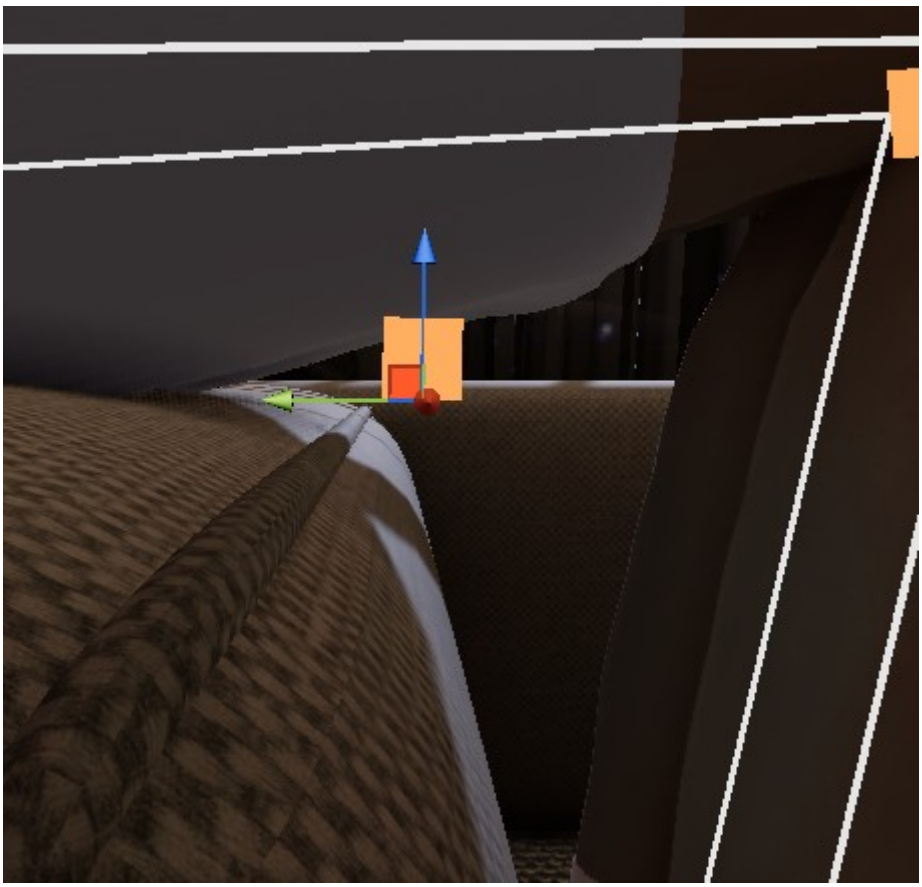
Climb posture itself is partial, it does not implement some poses related to low hanging legs due to different heights of main surface. There are two variants of climb posture: ClimbHigh – when target surface is closer to pelvis level, and ClimbLow – when it’s closer to knees. The difference is special smooth transition sequence when climbing high, and different legs angle when climbing low.

First of all switch game into custom pose mode, and perform “GoTo” action on Sofa POI. This action will teleport character to Sofa POI. Game detects current POI, but sometimes it may work wrong.

Then switch to Postures tool and “Apply” ClimbLow posture. After that action it is expected that character will be sitting in the air. Select root bone and drag it to edge of sofa surface.

Drag blue axis to align rootbone with sofa surface and green axis to match surface edge. Avoid dragging red axis, to prevent shifting during pose transition.

When dragging blue axis, keep an eye on body position. It is bad sign when puppet does not follow bones, it will end up with armature trembling and probably destabilize every pose in posture.



After you are fine with character's position, use "Implement" action of ClimbLow posture.

In the window you need to type missing part of ID. POI posture ID use this format: [Posture].[POI].[object of interest]. You can type here "Sofa".

> In most cases, when POI is bound to object of interest, you will end up with IDs like Sit.Table.Table. This in expected. The purpose of third part is to allow multiple targets at single POI, like Sit.Kitchen.Table/Sit.Kitchen.Sink or Sit.Sofa.Left/Sit.Sofa.Center.

Then exit custom pose mode, and relax character. Now, in poses menu you can see "Sofa" action. Click it. When character sits, its poses menu contains available poses and "Interrupt posture action". In this case it is called "Stand.Sofa".

To fix naming problems, transfer character to custom pose, and open "Edit descriptor" of new ClimbLow.Sofa.Sofa.

In this context, Display name is used for "Apply posture" action. You can type here "Sit on sofa".

Cancellation name is a "Cancel posture". You can type here "Stand up".

Orientation field is used for scenarios that require specific character orientation. Examples: Vanilla that provide different poses depending on character orientation.

Parent posture allows to nest postures. Nested posture can be applied only when character has parent posture. Example: Lie posture that require to Sit first.