more-sprites

New Super Mario Lost Worlds uses the more-sprites mod by Abood. While sprites in Newer replaced original sprites and actors from Nintendo, we can now simply expand the number of sprites.
An example of what the Newer Team did to add custom sprites can be found here.
If you want to add a sprite by replacing another, you have to change the function used by the game to create the sprite (*BuildBonePiece*), change the SpriteInfo (*UpdateBonePieceSpriteInfo*) and also change the list of files to be loaded for this sprite to work (3d models) (*BonePieceSpriteFileInfo*).

In NSMLW the yaml only contains the information which files to compile for the sprite, you can find the yaml here. The actual insertion of a sprite happens in the cpp files of the sprites now.

By the way: NSMBW has actors and sprites. Almost everything can be considered an actor but only those things that you can place in a level are sprites. There are lists for both and each sprite is an actor so you have to add two entries for each new sprite, one in the actor enum and one in the sprite enum.
There probably exists a more detailed explanation what's the difference between actors and sprites but this should suffice for the time being.

Since the Kamek version used in NSMLW copies all cpp files into one big file before compiling this one file, it's not actually needed to include something another file already included, but it can be considered to be good practice to include everything needed anyway.
For more-sprites you have to include the *profile.h* file:

```
#include <profile.h>
```

As in Newer, we still need a list of arc files to be loaded for each sprite:

```
const char* BonePieceNameList [] = { "lift_torokko", NULL };
```

You have to add *NULL* at the end since in C++ a string is saved as an array of chars and has no information about its own length. Most if not all functions know where a string ends by seeing a \0 or 0x00 byte — or simply *NULL*.

In Newer sprites have a build() function of its own type — e.g. *static daBonePiece_c *build()*, using more-sprites this function has to be changed to *dActor_c*:

```
class daBonePiece_c : public dStageActor_c {
    public:
        static dActor_c  *build();
        …
};
```

This change obviously has to be applied to the function itself as well and not just the definition of the function in the header of the class:

```
dActor_c  *daBonePiece_c::build() {
    void *buffer = AllocFromGameHeap1(sizeof(daBonePiece_c));
    daBonePiece_c *c = new(buffer) daBonePiece_c;
    return c;
}
```

Last but not least the sprite needs to actually be added to the game. This consists of two steps. First of all, you have to open profileid.h and add a new actor/profileid and a new sprite, preferably with the same name, at the bottom of both enums, but before *Num*.
For the sprite enum you would add **ANewSprite = 510,** here.
For the actors/profileid enum you would add **ANewSprite = 777,** here.

The last step now is creating entries the game uses to actually create the new sprites in-game:

```
const SpriteData BonePieceSpriteData =
    { ProfileId::BonePiece, 0, 0 , 0 , 0, 0x100, 0x100, 0, 0, 0, 0, 2};

Profile BonePieceProfile(&daBonePiece_c::build, SpriteId::BonePiece,
    BonePieceSpriteData, ProfileId::WM_BOSS_IGGY, ProfileId::BonePiece,
    "BonePiece", BonePieceNameList);
```

Honestly, I've got no clue how this works exactly. That's not really important though, as you can more or less simply copy this entirely and only change the ProfileId and SpriteId entries as well as the NameList at the end of the Profile.
Here an example for *ANewSprite* with the file list *ANewNameList* and the class *daNewSprite_c*:

```
const SpriteData ANewSpriteSpriteData =
    { ProfileId::ANewSprite, 0, 0 , 0 , 0, 0x100, 0x100, 0, 0, 0, 0, 2};

Profile ANewSpriteProfile(&daNewSprite_c::build, SpriteId::ANewSprite,
    ANewSpriteSpriteData, ProfileId::ANewSprite, ProfileId::ANewSprite,
    "NameOfThisSprite", ANewSpriteNameList);
```

Don't forget to change the SpriteData and Profile Objects name as to not overwrite another sprite before it even gets created in the game:
BonePieceSpriteData => ANewSpriteSpriteData & *BonePieceProfile => ANewSpriteProfile*

The rest really doesn't matter all that much. Maybe we will document this entirely at a later date.