How to Make a Custom Star Cutscene in Super Mario 64 DS

Type 14 Is Not a Mystery

By Josh65536

Table of Contents

[How Star Cutscenes Work 3](#_Toc491642299)

[Type 14: Star Cutscene Type IDs 3](#_Toc491642300)

[The Actual Cutscenes 4](#_Toc491642301)

[An Example 4](#_Toc491642302)

[A Custom Star Cutscene 5](#_Toc491642303)

[Instruction Types 6](#_Toc491642304)

Terms Used

**Hexit:** A hexadecimal digit

**Approach Factor:** A value that states the speed of exponential decay. Each frame, the distance to the target value is multiplied by 1 minus the approach factor.

**fxu:** Fixed-point units. The length of an orange brick block is exactly 200 fxu and Mario is about 144 fxu tall. 1 unit in the level editor is exactly 1000 fxu.

**Q12:** A number where the last 12 bits are fractional bits. 0x1000 represents 1 fxu.

**Horizontal angle:** The counterclockwise angle from the +z axis to the direction opposite the one the camera is facing.

**Stored angle:** An angle stored by the camera to be used for cutscenes.

**Stored vector:** A vector stored by the camera to be used for cutscenes.

**Stored Fix12:** A fixed-point number stored by the camera to be used for cutscenes.

**Stored short 0:** The first stored 16-bit integer in the camera. There is also a stored short 1.

**Field of view (FOV):** The vertical angle of camera viewing.

**FOV modifier:** A value that is multiplied by 27/128 and added to the final FOV.

# How Star Cutscenes Work

## Type 14: Star Cutscene Type IDs

In SM64DS, there is a Type 14 object in every level. SM64DSe shows the parameters in the Type 14 object, but they are in a pretty useless form. To convert them to a more useful form,

1. List the parameters in backward order. (eg. Snowman’s Land: 0 16 16 20)
2. Convert them to hexadecimal and give each number 2 hexits. (eg. 00 10 10 20)
3. Smush the hexits together. (eg. 00101020)

Now the star cutscene type IDs are listed in backward order, with the rightmost hexit being the one for Star 0 and the leftmost hexit being the one for star 7.

## The Actual Cutscenes

There is an array of 6 pointers to cutscene definitions at 0x020876e4 (EUR). The values (along with what the type IDs represent) are

Address Value

0x020876e4 0x02088388 (Type 0: Zoom in)

0x020876e8 0x020888f0 (Type 1: Zoom out)

0x020876ec 0x0208776c (Type 2: Stay still)

0x020876f0 0x020889b0 (Type 3: Spin)

0x020876f4 0x0208776c (Type 4: Stay still, identical to type 2)

0x020876f8 0x02088b14 (Type 5: Go crazy)

### An Example

Now, let’s take a look at the bytes of an actual cutscene definition, in this case, type 0, located at 0x02088388 (EUR).

**07** **04** **00 00** **00 00** 19

**0E** **04** **00 00** **0F 27** 12 00 00 7D 00 00 00 33

**07** **04** **00 00** **16 00** 1A

**10** **04** **17 00** **0F 27** 14 F0 00 66 00 10 05 FF FF 08

**0B** **04** **17 00** **FF FF** 04 71 1C 3E 01

**0D** **04** **1E 00** **1E 00** 18 00 03 30 00 00 80

**00**

The cutscene definition is a sort of scripting language (Yay, a scripting language in SM64DS). It contains multiple instructions each made of a variable number of bytes. Each frame the star cutscene is performed, every instruction is processed.

The colors above are supposed to separate different-sized fields in the instructions. The bytes are in little endian order, so each field has its bytes backwards. If a 2-byte field says 17 00 for example, it really means 0x0017.

Everything in bold is something that each instruction has in common.

* The 1st (green) field is the length of the instruction in bytes, including itself. The cutscene definition ends where the instruction length is 0.
* The 2nd (yellow) field is the type of instruction to execute. 4, for example, means do something with the camera.
* The 3rd and 4th fields (red and cyan, respectively) are the mintime and maxtime, respectively, of the instructions.

An instruction is executed if and only if the frame counter for the cutscene is at least the mintime and at most the maxtime, or it’s at least the mintime and the maxtime is negative and the cutscene has not ended (2’s complement).

After that comes specific bytes for the instruction type. All instructions of type 4, for example, share a field (non-bold green) that says what to do with the camera.

* For example, the code 0x12 adjusts the camera’s focus point (also known as look-at position) to a certain position relative to its owner (in this case, the player owns the camera) using exponential decay (instead of constant speed).
  + The next three fields list the X (red), Y (yellow), and Z (cyan) offset coordinates of the target look-at position in fxu.
  + The last field (pink) states the approach factor multiplied by 256.

So what does this specific example do?

* On frame 0, some level-specific star-specific stuff are done (Please don’t use code 0x19 on instruction type 4.)
* From frame 0 to frame 9,999 (inclusive) (way longer than the cutscene), the camera’s look-at position approaches (0, 125, 0) fxu from the player. The approach factor is 1/5
* From frame 0 to frame 22 (inclusive), the camera’s position approaches a distance of 800 fxu (approach factor = 1/20), 33.75° above the player (approach factor = 1/16), with the horizontal angle being the stored angle (approach factor = 1/8). Code 0x1a is hardcoded to have this exact behavior.
* From frame 23 to frame 9,999 (inclusive), the camera’s position approaches a distance of 240 fxu (approach factor = 2/5), 22.5° above the player (approach factor = 1/5), with the horizontal angle being the stored angle (approach factor = 1/8).
* From frame 23 until the cutscene ends, the camera’s FOV modifier approaches 0x1c71 (increasing the field of view by about 16.875°) with a speed of 0x13e per frame.
* On frame 30, the camera’s FOV modifier is set to 0x300. It then decreases with a speed of 0x3 while oscillating with an angular speed of 180° per frame.
* Remember, the definition ends with an instruction that claims to have 0 length.

# A Custom Star Cutscene

Now, let’s try to make a star cutscene that spins around the player, then zooms in when the player thrusts the star in the air, then zooms out.

First, go to 0x876f8 in the ROM and replace the bytes there with 00 6c 00 02 so star cutscene type 5 points to 0x02006c00. (It is not recommended to replace star cutscene type 6 (nonexistent) and above). Then go to 0x6c00 in the ROM (empty unused space) and let’s type the cutscene bytes!

First, we want to spin the camera around the player, so we use code 0x16 for instruction type 4. Let’s do it from frame 0 to frame 22 (inclusive). The distance speed should be 0 and the vertical angular speed should by 0°, and let’s set the horizontal angular speed to 360°/23 so it makes a full rotation. To convert degrees to the integer to write, multiply by 65536/(360°). Remember to type the bytes backwards because little endian is used. In this case, the integer is 0xb21.

**0F 04 00 00 16 00** 16 00 00 00 00 00 00 21 0B

Then, let’s zoom the camera in. Let’s copy the zoom in from star cutscene type 0. It consists of the camera getting up close and personal to the player and an FOV modification. Because we then want to zoom out the camera, change the maxtime for the first instruction here to 59.

**10** **04** **17 00** **3B 00** 14 F0 00 66 00 10 05 FF FF 08

**0B** **04** **17 00** **FF FF** 04 71 1C 3E 01

Let’s make sure to add in the shake when the player thrusts the star in the air. Copy it from star cutscene type 0.

**0D** **04** **1E 00** **1E 00** 18 00 03 30 00 00 80

Finally, for the zoom out, use code 0x16 for instruction type 4, and increase the distance by 65 fxu each frame, starting from frame 60. Note to multiply 65 fxu by 0x1000 to get the integer to type in, and remember to reverse the bytes.

**0F 04 3C 00 0F 27** 16 00 10 04 00 00 00 00 00

And to finish off the cutscene definition, type in 0

**00**

And now you can test it out!

Appendix: The Cutscene Script Documentation

## Instruction Types

0x04: Control the camera

Offset 0x06 (size 1): Function to use

0x00: Set camera look-at position

Offset 0x07 (size 2): Look-at position X in fxu

Offset 0x09 (size 2): Look-at position Y in fxu

Offset 0x0b (size 2): Look-at position Z in fxu

0x01: Set camera position

Offset 0x07 (size 2): Position X in fxu

Offset 0x09 (size 2): Position Y in fxu

Offset 0x0b (size 2): Position Z in fxu

0x02: Set camera look at position and camera position

Offset 0x07 (size 2): Look at position X in fxu

Offset 0x09 (size 2): Look at position Y in fxu

Offset 0x0b (size 2): Look at position Z in fxu

Offset 0x0d (size 2): Position X in fxu

Offset 0x0f (size 2): Position Y in fxu

Offset 0x11 (size 2): Position Z in fxu

0x03: Set FOV modifier

Offset 0x07 (size 2): FOV modifier

0x04: Adjust FOV modifier

Offset 0x07 (size 2): Target FOV modifier

Offset 0x09 (size 2): Speed

0x05: Adjust screen size from full size to target values (uses minTime and maxTime for gradient)

Offset 0x07 (size 1): Screen left

Offset 0x08 (size 1): Screen bottom

Offset 0x09 (size 1): Screen right

Offset 0x0a (size 1): Screen top

0x06: Weird cubic interpolation?

Offset 0x07 (size 4): ??? (pointer)

Offset 0x0b (size 4): ??? (pointer)

<Hardcoded cutscene stuff here>

0x0d: Set stored Fix12

Offset 0x07 (size 4): New value (Q12)

0x0e: Adjust stored Fix12

Offset 0x07 (size 4): Target value (Q12)

Offset 0x0b (size 4): Speed

0x0f: Adjust camera look-at position (exponential decay)

Offset 0x07 (size 2): Target look-at position X in fxu

Offset 0x09 (size 2): Target look-at position Y in fxu

Offset 0x0b (size 2): Target look-at position Z in fxu

Offset 0x0d (size 2): Approach factor X

Offset 0x0f (size 2): Approach factor Y

Offset 0x11 (size 2): Approach factor Z

0x10: Adjust camera position (exponential decay)

Offset 0x07 (size 2): Target position X in fxu

Offset 0x09 (size 2): Target position Y in fxu

Offset 0x0b (size 2): Target position Z in fxu

Offset 0x0d (size 2): Approach factor X

Offset 0x0f (size 2): Approach factor Y

Offset 0x11 (size 2): Approach factor Z

0x11: Set stored angle toward pause view position, set stored shorts to 0.

0x12: Adjust camera look-at position to offset from owner (exponential decay)

Offset 0x07 (size 2): Target look-at offset X in fxu

Offset 0x09 (size 2): Target look-at offset Y in fxu

Offset 0x0b (size 2): Target look-at offset Z in fxu

Offset 0x0d (size 1): Approach factor times 256

0x13: Adjust camera look at position to offset from owner rotated by owner’s facing angle (exponential decay)

Info:

Offset 0x07 (size 2): Target look-at offset X in fxu

Offset 0x09 (size 2): Target look-at offset Y in fxu

Offset 0x0b (size 2): Target look-at offset Z in fxu

Offset 0x0d (size 1): Approach factor times 256

0x14: Adjust camera position and angles relative to owner position (exponential decay)

Offset 0x07 (size 2): Target distance in fxu

Offset 0x09 (size 1): Approach factor of distance times 256

Offset 0x0a (size 2): Target vertical angle

Offset 0x0c (size 1): 1 / Approach factor of vertical angle (set to 0 to not change vertical angle)

Offset 0x0d (size 2): Target horizontal angle (set to -1 to use CAMERA->targetAngle)

Offset 0x0f (size 1): 1 / Approach factor of horizontal angle (set to 0 to not change horizontal angle)

0x15: Spin camera look at position around camera position (Sets stored vector to look-at position on first frame)

Offset 0x07 (size 2): Vertical angular speed

Offset 0x09 (size 2): Horizontal angular speed

0x16: Move/spin camera position around owner position

Info:

Offset 0x07 (size 4): Distance speed

Offset 0x0b (size 2): Vertical angular speed

Offset 0x0d (size 2): Horizontal angular speed

0x17: Adjust FOV modifier (exponential decay)

Offset 0x07 (size 2): Target FOV modifier

Offset 0x09 (size 1): 1 / Approach factor

0x18: Adjust FOV modifier (works only if new value > old value)

0x07 (size 2): New FOV modifier value

0x09 (size 2): FOV speed toward 0

0x0b (size 2): FOV oscillation angular speed

0x19: Does nothing

0x1a: Approach a distance of 800 fxu (approach factor = 1/20), 33.75° above the player (approach factor = 1/16), with the horizontal angle being the stored angle (approach factor = 1/8).

0x1b: Set camera look at position and camera position to rotated offset from owner

Offset 0x07 (size 2): Target look at offset X in fxu

Offset 0x09 (size 2): Target look at offset Y in fxu

Offset 0x0b (size 2): Target look at offset Z in fxu

Offset 0x0d (size 2): Target offset X in fxu

Offset 0x0f (size 2): Target offset Y in fxu

Offset 0x11 (size 2): Target offset Z in fxu

<Hardcoded cutscene stuff up to 0x26>

0x05: Something to do with WiFi

0x06: Change cutscene script

Offset 0x06 (size 4): Address of new cutscene script

0x07: Change music

Offset 0x06 (size 4): New music ID

0x08: Play sound from sequence archive 1 (Player voices)

Offset 0x06 (size 4): Sound ID

0x09: Play sound from sequence archive 2 (System)

Offset 0x06 (size 4): Sound ID

0x0a: Display message

Offset 0x06 (size 2): Raw message ID

0x0b: Change level

Offset 0x06 (size 1): New level ID

Offset 0x07 (size 1): Entrance ID

Offset 0x08 (size 1): Star ID

Offset 0x09 (size 4): Cutscene address (or 0 if no cutscene)

0x0c: Fade to white

0x0d: Fade from white

0x0e: Fade to black

0x0f: Fade from white (note: broken)

0x10: Fade to black, then fade from black

0x11: Sets int at 0x02110aec to 0