Mode 7

Aditya Patel

Introduction

In the world of video games, player experience and game design have always been a priority for developers. One such experience and design gamechanger was Mode 7, used by the Super Nintendo Entertainment System (SNES) during the nineties. Mode 7 introduced a new dimension of visuals to create immersive environments and visual effects to define an era of nostalgia for gamers.

### What Is Mode 7?

Mode 7 is one of eight graphics modes of the SNES, that allowed for scaling, rotation and transformation of a background layer [1]. This provided the illusion of a 3D environment in the 2D plane. Because true 3D graphics were beyond the computational and processing abilities of gaming consoles at the time, Mode 7 provided a resource efficient method to simulate this 3D space. This offers a captivating visual experience without demanding a large amount of computational power.

The gaming legacy of Mode 7 is reflected in a group of gamers that fondly remember the 16-bit graphics of games such as F-Zero and Super Mario Kart on the SNES console, and further inspiring some modern-day games to follow a similar aesthetic in development – such as the wildly popular Octopath Traveler series. The technological legacy of Mode 7 is the inspiration for developers to push the envelope of consoles to generate the latest and greatest in graphical technology, leading to experimentation with 3D modeling, and culminating in dedicated 3D hardware in modern consoles.

### What Are Fixed Point Numbers?

Mode 7 heavily relies on the use of fixed-point arithmetic. Fixed-point numbers represent real numbers with a fixed number of digits after the decimal or binary point. Even though floating-point numbers can represent a larger set of values at the expense of computational resources, fixed-point numbers offer a computationally simpler alternative at the expense of range, which is suitable in applications in legacy technology – like the SNES.

In a fixed-point system, numbers are represented by a fixed number of integer and fractional bits – for example, an 8-bit fixed-point number would use 4 bits for the integer and 4 bits for the fractional part. In the computer, this would be represented as:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Integer |  |  | Fraction |  |  |

Figure 1: Memory Allocation of an 8-bit Fixed-Point Number.

Now, taking an example for the above 8-bit, signed, fixed-point number, with a 4 and 4 split between the integer and the fraction, we determine the range for the integer part as being distinct values, from to , in this case, with n= 4, from

The fractional also takes a similar process, with the bits reflected as , therefore the smallest value that can be represented is, with the maximum fractional part valuation at , which makes our range, with combined fractional and integer parts, . The smallest step between values is determined by the smallest increment of the fractional part, which is 1/16, or 0.0625.

### Converting To Fixed-Point

For an example of converting a number to this fixed-point format, using the value of   
-5.375, we do the following [2]:

1. Convert the integer part

* Since we’re picking a point that’s more negative than 5, Use the 2’s complement to find -6 in binary:
  + Positive value: 6 = 0110
  + Invert: 1001
  + Add 1: 1010
* The integer value is 1010

1. Convert the fractional part

* Take 1 – 0.375 = 0.625 as the decimal
* Validate:

1. Combine the parts

* 1010.1010

To verify:

1. Convert to positive using 2’s complement:

* Subtract 0000.0001: 1010.1001
* Invert bits: 0101.0110

1. Convert to Decimal

* Integer: 0101 = 5
* Fractional 0.0110 = 0.375
* 5 + 0.375 = 5.375, which is correct.

### Using Fixed-Point Arithmetic to Perform Scaling

As stated before, Mode 7 is the eighth graphical mode available for use by the SNES console. The SNES organizes its background modalities by the number of background layers and the number of colors that can be displayed [3].

A black rectangular table with white text

Description automatically generated

Figure 2: SNES Video Modes and Background Layers

|  |  |
| --- | --- |
| A video game screen with a monster and a person in a helmet  Description automatically generated | A video game screen with cars and buildings  Description automatically generated |

Figure 3: Comparison of Mode 0(left) to Mode 7(right) graphics. Note that mode 0 visually looks flat, compared to the cars in mode 7 that appear to have three dimensions.

Mode 0 was the most common modality, used for titles such as Super Metroid, and Mode 2 is interesting since it was able to output two background layers capable of the same number of colors that were able to scroll independently of each other, which worked best for games in the   
“beat ‘em up” genre. What makes Mode 7 so unique and interesting, however, is the ability for it to utilize scaling and rotation to simulate 3D effects.

Matrix operations are used on fixed-point numbers to shift the screen coordinates with respect to the background coordinates using affine transformations The 2D transformation is specified for each scanline by 6 parameters – a, b, c, d, x0, and y0. The first four parameters, a, b, c, and d, define the M matrix, which is the augmentation matrix. The other two parameters, x0 and y0, define the origin point. This transformation solves the linear system of equations:

or in matrix notation:

This matrix transformation arithmetic in Mode 7 is carried out on 16-bit signed fixed-point integers.

### References

[1] "Mode 7," Wikipedia. [Online]. Available: <https://en.wikipedia.org/wiki/Mode_7>.

[2] A. Kartik, "Converting fixed-point to floating-point format and vice versa," Medium, Jun. 20, 2018. [Online]. Available: <https://medium.com/incredible-coder/converting-fixed-point-to-floating-point-format-and-vice-versa-6cbc0e32544e>.

[3] Retro Game Mechanics Explained, "Mode 7: How the SNES warped a generation," YouTube, Sep. 25, 2019. [Online]. Available: https://www.youtube.com/watch?v=kwAjG2dj5ZM.