PPU的原理是这样的：它有2KB的显存，用来存储背景和精灵的图案数据。它还有一个内部寄存器，用来控制显示参数，比如水平和垂直滚动位置、屏幕模式、中断信号等。PPU每隔一定时间（约为16.7毫秒）会向CPU发送一个中断信号，告诉CPU可以更新显存和寄存器的内容了。这个时间点叫做垂直空白期（VBlank），因为此时电视机不在显示画面，而是准备刷新下一帧[3](http://182.61.19.91/2022/04/08/%E7%9F%A5%E9%81%93fc%E4%B9%9F%E7%9F%A5%E9%81%93%E4%BB%BB%E5%A4%A9%E5%A0%82%EF%BC%8C%E9%82%A3%E8%BF%99%E4%B8%AAppu%E6%98%AF%E4%BB%80%E4%B9%88%EF%BC%9F/" \t "_blank)。

在垂直空白期之外，PPU会根据显存和寄存器的内容，逐行逐像素地生成复合视频信号，并输出到电视机上。复合视频信号包含了亮度和色度两个分量，分别决定了像素点的明暗和颜色[3](http://182.61.19.91/2022/04/08/%E7%9F%A5%E9%81%93fc%E4%B9%9F%E7%9F%A5%E9%81%93%E4%BB%BB%E5%A4%A9%E5%A0%82%EF%BC%8C%E9%82%A3%E8%BF%99%E4%B8%AAppu%E6%98%AF%E4%BB%80%E4%B9%88%EF%BC%9F/" \t "_blank)。PPU会根据调色板来选择每个像素点的颜色，调色板是一个32字节的内部SRAM，可以存储48种颜色中的任意16种[1](https://www.zhihu.com/question/20651716" \t "_blank) [2](https://zhuanlan.zhihu.com/p/418078067" \t "_blank)。

背景是由许多8x8像素的方块（称为tile或cell）组成的，每个方块可以有4种颜色之一。方块的排列方式由另外一个2KB的显存（称为name table）来决定。name table可以看作是一个32x30的矩阵，每个元素对应一个方块编号[1](https://www.zhihu.com/question/20651716" \t "_blank) [2](https://zhuanlan.zhihu.com/p/418078067" \t "_blank)。

精灵是指游戏中能够移动或变化的图形元素，比如角色、敌人、子弹等。精灵也是由8x8或8x16像素的方块组成的，但是它们不受name table限制，可以在屏幕上任意位置显示。每个精灵可以有4种颜色之一，并且可以水平或垂直翻转[1](https://www.zhihu.com/question/20651716" \t "_blank) [2](https://zhuanlan.zhihu.com/p/418078067" \t "_blank)。

PPU最多可以同时显示64个精灵，并且每条扫描线最多只能显示8个精灵。如果超过这个限制，就会出现闪烁或消失的现象[1](https://www.zhihu.com/question/20651716" \t "_blank) [2](https://zhuanlan.zhihu.com/p/418078067" \t "_blank)。

PPU的2KB显存是这样划分的：

* $0000-$07FF：用来存储背景和精灵的图案数据，每个图案占用16字节，共有256个图案[1](https://www.zhihu.com/question/20651716" \t "_blank) [2](https://www.nesdev.org/wiki/PPU_memory_map" \t "_blank)。
* $0800-$0FFF：用来存储另一组背景和精灵的图案数据，也是256个图案[1](https://www.zhihu.com/question/20651716" \t "_blank) [2](https://www.nesdev.org/wiki/PPU_memory_map" \t "_blank)。
* $2000-$23FF：用来存储第一个name table，即背景方块的排列方式，每个方块占用一个字节，共有960个方块[1](https://www.zhihu.com/question/20651716" \t "_blank) [2](https://www.nesdev.org/wiki/PPU_memory_map" \t "_blank)。
* $2400-$27FF：用来存储第二个name table，同上[1](https://www.zhihu.com/question/20651716" \t "_blank) [2](https://www.nesdev.org/wiki/PPU_memory_map" \t "_blank)。
* $2800-$2BFF：用来存储第三个name table，同上[1](https://www.zhihu.com/question/20651716" \t "_blank) [2](https://www.nesdev.org/wiki/PPU_memory_map" \t "_blank)。
* $2C00-$2FFF：用来存储第四个name table，同上[1](https://www.zhihu.com/question/20651716" \t "_blank) [2](https://www.nesdev.org/wiki/PPU_memory_map" \t "_blank)。
* $3000-$3EFF：镜像区域，与$2000-$2EFF相同[1](https://www.zhihu.com/question/20651716" \t "_blank) [3](https://blog.csdn.net/qq_34254642/article/details/104193445" \t "_blank)。
* $3F00-$3F0F：用来存储背景调色板索引，每种颜色占用一个字节，共有16种颜色[1](https://www.zhihu.com/question/20651716" \t "_blank)
* $3F10-$3F1F：用来存储精灵调色板索引，每种颜色占用一个字节，共有16种颜色[1](https://www.zhihu.com/question/20651716" \t "_blank) ^
* $3F20-$3FFF：镜像区域，与$3F00-$3F1F相同^

这个游戏是使用的水平卷动方式，即垂直排列的两个name table，然后将$2800和$2C00镜像到$2000和$2400[1](https://www.bilibili.com/video/BV1ms411p7ea/" \t "_blank) [3](https://www.zhihu.com/question/24515643" \t "_blank)。这样可以实现上下方向的无缝切换，而左右方向则是固定不变的。这个游戏共有32关，每关都有不同的敌机和BOSS[1](https://www.bilibili.com/video/BV1ms411p7ea/" \t "_blank) [2](https://www.bilibili.com/video/BV1MA41177BV/" \t "_blank)。

nametable是从.nes文件的CHR-ROM区域更新数据的。.nes文件是FC游戏的存储格式，它包含了PRG-ROM和CHR-ROM两个部分，分别对应CPU和PPU的可读内存[1](https://austinmorlan.com/posts/nes_rendering_overview/" \t "_blank) [2](https://www.nesdev.org/wiki/PPU_nametables" \t "_blank)。PRG-ROM存储了游戏的程序代码和数据，CHR-ROM存储了游戏的图案数据，即背景和精灵的8x8像素方块[1](https://austinmorlan.com/posts/nes_rendering_overview/" \t "_blank) [2](https://www.nesdev.org/wiki/PPU_nametables" \t "_blank)。

PPU通过一些寄存器来控制nametable的更新。其中最重要的是$2000寄存器，它可以指定当前使用哪个nametable（$2000、$2400、$2800或$2C00），以及是否启用水平或垂直卷动[1](https://austinmorlan.com/posts/nes_rendering_overview/" \t "_blank) [3](https://www.copetti.org/zh-hans/writings/consoles/nes/" \t "_blank)。另外还有$2005和$2006寄存器，它们可以指定当前在nametable中的位置（X和Y坐标），以及要写入nametable中的值[1](https://austinmorlan.com/posts/nes_rendering_overview/" \t "_blank) [3](https://www.copetti.org/zh-hans/writings/consoles/nes/" \t "_blank)。

当CPU要更新nametable时，它需要先向PPU发送一些命令，比如设置卷动方式、选择图案表地址、指定要写入的位置等等。然后CPU就可以通过$2007寄存器向PPU发送图案数据，这些数据会被写入到相应的nametable中，并且自动增加X或Y坐标[1](https://austinmorlan.com/posts/nes_rendering_overview/" \t "_blank) [3](https://www.copetti.org/zh-hans/writings/consoles/nes/" \t "_blank)。

例如，在1942游戏中，当玩家按下开始键进入第一关时，CPU会向PPU发送以下命令：

* 向$2000寄存器写入10000000（二进制），表示选择第一个name table（$2000）并启用垂直卷动。
* 向$2006寄存器写入00100000（二进制），表示要写入到name table中的高8位地址。
* 向$2006寄存器写入00000000（二进制），表示要写入到name table中的低8位地址。
* 向$2007寄存器连续写入960个字节（从00到3B），表示填充整个name table为全黑色背景。
* 向$2006寄存器写入00100011（二进制），表示要写入到attribute table中的高8位地址。
* 向$2006寄存器写入11000000（二进制），表示要写入到attribute table中的低8位地址。
* 向$2007寄存器连续写入64个字节（从00到3F），表示填充整个attribute table为全黑色背景。