Optimización de memoria en videojuegos.

Las consolas y computadoras actuales superan a los equipos de hace unos años atrás, por lo tanto el buen uso de recursos era un factor determinante para que un videojuego pudiese o no ser corrido por una consola.

Con tantas limitaciones de hardware era necesario pensar en grandes soluciones, precisamente el mostrar los gráficos del videojuego eran un problema debido a la cantidad de polígonos que podía tener el elemntos modelado para videojuegos 3D. De igual forma al juegos 2D como Pokémon se optaba por utilizar una cantidad reducida de sprites los cuales, de ser necesario se invierten para lograr obtener de una image viendo hacia la izquierda a una viendo a la derecha y no tener que almacenar más imágenes, esto era útil para simular que el personaje caminaba ya que solo se intercalaba entre la original y la invertida para lograr una “animación” de movimiento. De igual se trabajaba con elementos que requerían muchos polígonos (como esferas) donde se sustituyen modelos tridimensionales por sprites que quedaban de frente al jugador, de esta manera parecía que se tenía una esfera.

Algo que me parece aún más sorprendente es el hecho de que se dibujan y “borraban” secciones de los escenarios, como en Pokemon, donde si el escenario no cabía en la pantalla simplemente no se dibujaba y se prefiere dejar de cargarlo, aplicando como lógica el hecho de que si no se puede ver entonces no está el elemento. Pero luego de hablar un poco sobre trucos utilizados en diversos videojuegos como el uso de sprints invertidos, sprites para generar elementos que requieren de muchos polígonos etc. hubieron dos cosas que me llamaron la a atención, una es de Pokemon respecto a los escenarios ya que me recordó a cuando yo estaba dibujando la letra “E” con líneas, según lo que investigue los elementos del mapa están conformados por bloques de 4 elementos, es decir, en lugar de ser un bloque para un árbol, otro para un cartel y otros para elementos individuales, se tenía un bloque de cuatro que tenía un árbol, un cartel, un arbusto y solamente césped, el motivo de esto es que si se tenían 20 casillas de alto y 20 de ancho, no tuvieran que programarse 400 casillas, sino solamente usa 10 casillas por 10 casillas teniendo que programar únicamente 100 para lograr cargar el mapa.

Y lo segundo que me llamo la atencion fue algo que yo consideraría un error, o algo que no está bien hecho pero sin embargo “funciona” en el Zelda de NES, donde el número máximo de rupias a almacenar eran 255, número más grande que se puede almacenar un solo byte de memoria que no estoy seguro si no se agrego un byte más porque de verdad no podían utilizar uno más, pero algo muy curioso que es a lo que yo me refiero con que para mi eso es un error, es el hecho de que al utilizar flechas la forma de saber que se había gastado una no se veía mediante un contador de flechas, no existía un contador de flechas, por lo que al utilizar una se reducía el acumulado de rupias en 1, es decir que el juego utiliza el mismo byte para almacenar dos cosas distintas.