Vircon32

CONSOLA VIRTUAL DE 32 BITS

Especificación del sistema

Parte 4: El chip gráfico (GPU)

Documento con fecha 2023.12.27 Escrito por Carra

¿Qué es esto?

Este documento es la parte número 4 de la especificación del sistema Vircon32. Esta serie de documentos define el sistema Vircon32, y provee una especificación completa que describe en detalle sus características y comportamiento.

El principal objetivo de esta especificación es definir un estándar de lo que es un sistema Vircon32, y cómo debe implementarse un sistema de juego para que se considere conforme a él. Además, al ser Vircon32 es un sistema virtual, un importante objetivo adicional de estos documentos es proporcionar a cualquiera el conocimiento para crear sus propias implementaciones de Vircon32.

Sobre Vircon32

El proyecto Vircon32 fue creado de forma independiente por Carra. El sistema Vircon32 y su material asociado (incluyendo documentos, software, código fuente, arte y cualquier otro elemento relacionado) son propiedad del autor original.

Vircon32 es un proyecto libre y de código abierto en un esfuerzo por promover que cualquiera pueda jugar a la consola y crear software para ella. Para obtener información más detallada al respecto, se recomienda consultar los textos de licencia incluidos en cada uno de los programas disponibles.

Sobre este documento

Índice

La parte 4 de la especificación define el chip gráfico (GPU) de la consola. Este documento describirá el comportamiento de este chip, sus puertos de control, los comandos de dibujo que proporciona y el proceso que usa para producir la salida de video.

1 Introducción

La GPU es el chip de video de la consola Vircon32. Es responsable de dibujar todo lo que se ve en la pantalla. Para ello usará las texturas contenidas tanto en la BIOS como en el cartucho y les aplicará efectos gráficos básicos.

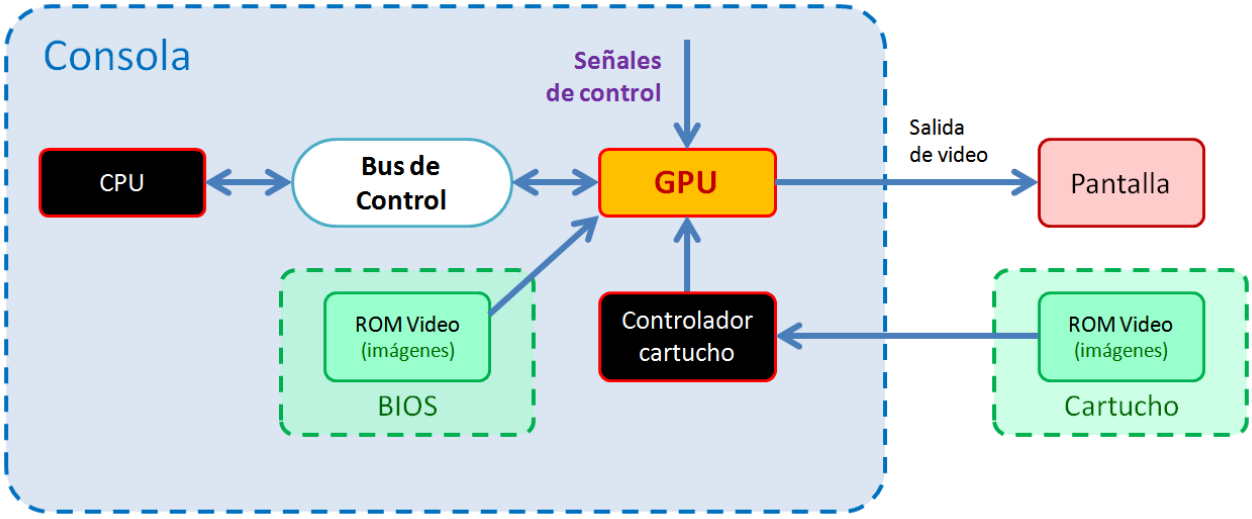
Este chip gráfico está muy simplificado: sólo hay 2 tipos de acciones que la GPU puede realizar para dibujar en pantalla:

- Puede borrar la pantalla con un color constante.
- Puede dibujar en pantalla una región de una de las texturas disponibles.

Ambas se ejecutarán como comandos de dibujo a petición del procesador. Aparte de esto, hay algunos efectos gráficos que pueden aplicarse a estas acciones. Éstos se tratarán en secciones posteriores.

2 Conexiones externas

Como la GPU es sólo uno de los chips que forman la consola, no puede funcionar aislado. Esta figura muestra todas las comunicaciones de la GPU con otros componentes. Como se ve, la GPU tiene conexiones con todas las ROMs de video disponibles para poder usar las imágenes que contienen como texturas. Obsérvese que, por claridad, los diagramas de la consola de la parte 2 de la especificación omiten a propósito estas conexiones.



Cada una de estas conexiones se explicará individualmente en las secciones siguientes.

2.1 Señales de control

Como todos los componentes de la consola, la GPU recibe las señales de reset, frame y ciclo. Las respuestas a esas señales se detallan en la sección 12 de este documento.

2.2 Bus de Control

La GPU está conectada como dispositivo esclavo al Bus de Control, con ID de dispositivo = 2. Esto permite al maestro del bus (la CPU) pedir operaciones de lectura o escritura en los puertos de control que expone la GPU. La lista de puertos de la GPU y sus propiedades se detallarán en secciones posteriores.

2.3 Chip de la BIOS

La BIOS, que siempre está presente, contiene una ROM de video con exactamente 1 imagen. La GPU puede acceder a esa imagen a través de su canal de textura con ID = –1.

2.4 Controlador de cartucho

Al igual que con la BIOS, la GPU puede acceder a cualquier imagen presente en el cartucho a través de canales de textura con IDs de 0 a 255. Pero la conexión debe usar como proxy el controlador de cartucho, ya que puede no haber cartucho presente. Incluso si hay cartucho, cada uno puede tener un número distinto de imágenes (de 0 a 256).

2.5 Pantalla

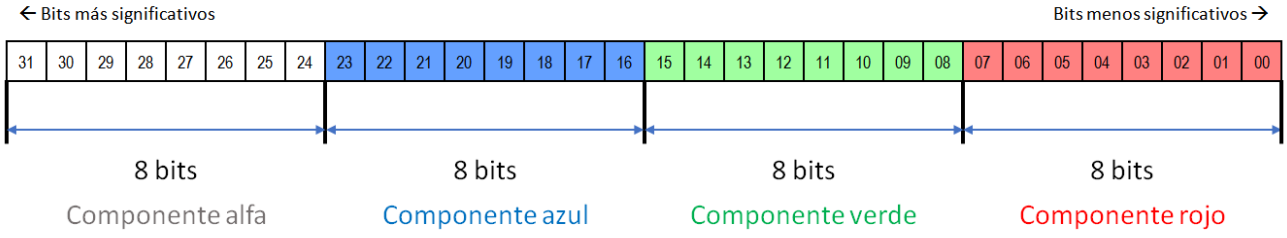
En cada frame, una vez que se ha terminado de dibujar, la GPU debe ser capaz de enviar el resultado a la pantalla para que pueda verse. Esto se hace a través de la conexión de salida de video. En la sección 11 se explica cómo.

3 Conceptos funcionales

Antes de explicar las funciones gráficas de la GPU o las variables internas que les afectan, debemos presentar algunos conceptos básicos en los que se basa el diseño de la GPU.

3.1 Pixels y colores

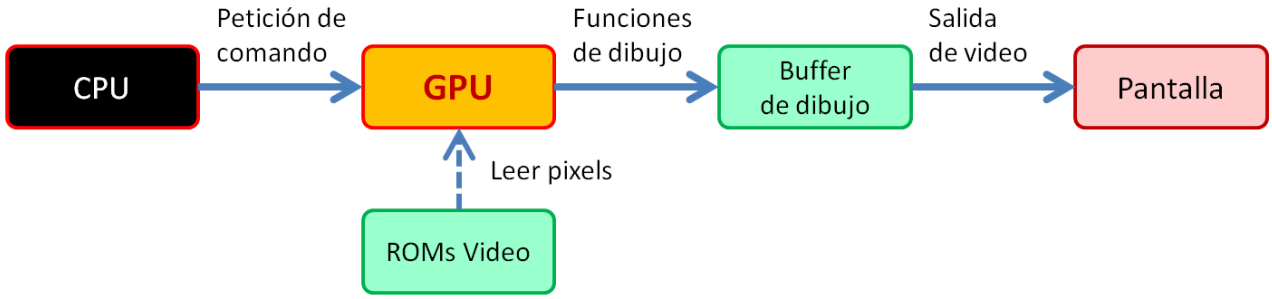
La unidad mínima de información de video que maneja la GPU es el pixel, que representa una única posición discreta en la pantalla o en una textura. Cada pixel se expresa como un único color RGBA de 32 bits, como se documentó en la parte 2 de la especificación:



3.2 El buffer de dibujo

El buffer de dibujo es una matriz 2D rectangular de pixels, del mismo tamaño que la pantalla (640 x 360 pixels). Cada pixel individual se identifica por su posición desde la esquina superior izquierda, es decir, desde (0, 0) hasta (639, 359).

La GPU utiliza este buffer como sustituto de la propia pantalla: el contenido del buffer de dibujo se va modificando con cada operación gráfica de la GPU. Luego, en cada nuevo frame, la señal de salida de video se actualiza con el contenido actual del buffer de dibujo y eso hace que cualquier cambio desde el último frame se haga visible. En conjunto, la operación de la GPU sobre el buffer de dibujo sigue este esquema:



Cabe destacar que el componente alfa de un pixel sólo necesita usarse durante las operaciones de dibujo, para el proceso de mezcla de colores en el buffer de dibujo. Después de esto, el buffer de dibujo no necesita guardar el componente alfa de sus pixels. Depende de la implementación decidir si el formato de pixel para el buffer de dibujo usa 32 bits o sólo 24 bits omitiendo el alfa. En cualquier caso, se puede suponer que los pixels del buffer de dibujo tienen siempre opacidad total (alfa = 255).

3.3 Texturas de la GPU

Una textura de GPU es una matriz 2D con un tamaño fijo de 1024 x 1024 pixels. Las coordenadas comienzan en la esquina superior izquierda, que es el pixel (0, 0). Salvo borrar la pantalla, todas las funciones de dibujo necesitan usar una textura para dibujar.

La GPU identifica y accede a las texturas disponibles a través de una matriz de canales de textura numerados. Hay 256 canales utilizables por el cartucho (con IDs de textura de 0 a 255), y un canal adicional con ID = –1 para la textura de la BIOS.

3.4 Conexión a ROMs de video

La GPU en sí misma no contiene texturas, así que necesita leer las imágenes guardadas en la BIOS y los cartuchos conectándose a sus respectivas ROMs de video. Una ROM de video es una región de memoria de sólo lectura que contiene una secuencia de imágenes. Cada una de estas imágenes puede tener cualquier anchura y altura posibles, desde 1 x 1 pixels hasta el tamaño de una textura de GPU.

Cuando se conecta un cartucho que contiene N imágenes, los N primeros canales a partir de la ID = 0 se asignan a cada una de las imágenes, por orden. El resto de canales hasta la ID = 255 quedarán sin usar y no serán accesibles. El canal con ID = -1 se asigna a la imagen de la BIOS, que se garantiza que existe y es única.

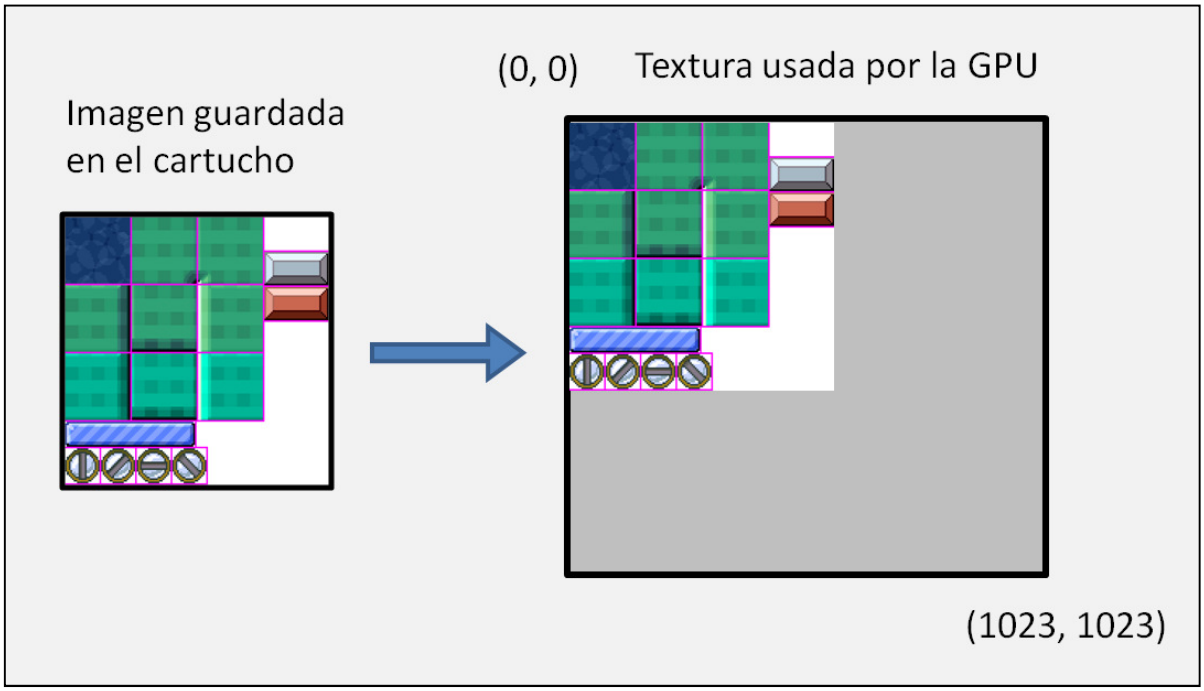
Este proceso de conexión ocurre cada vez que se inserta un nuevo cartucho. Aún así, en sistemas Vircon32, los cartuchos sólo pueden insertarse con la consola apagada. Por lo tanto, en las implementaciones, los procedimientos necesarios para que la GPU establezca esta conexión se pueden retrasar hasta el siguiente encendido de la consola.

Depende de la implementación decidir cómo establecer conexión con las ROMs de video, y localizar y leer sus imágenes. Por ejemplo, se puede leer todas las imágenes de las ROMs de video de antemano al establecer la conexión. Otra opción sería mantener punteros a las imágenes y hacer que la GPU lea los valores de los pixels sobre la marcha.

Extensión de las imágenes

Las texturas de GPU tienen un tamaño fijo, pero los canales leen sus pixels de imágenes que pueden tener distintos tamaños. Para resolver esto, cada canal adaptará su imagen asignada para que pueda usarse como textura de GPU. Se ampliarán automáticamente hasta 1024 x 1024, considerando el resto de pixels vacíos (con sus 4 componentes 0). Se extiende por la parte inferior derecha para conservar las coordenadas de los pixels.

Esta imagen muestra cómo una imagen guardada en una ROM de video se extendería con pixels vacíos (aquí se muestra en color gris por claridad).

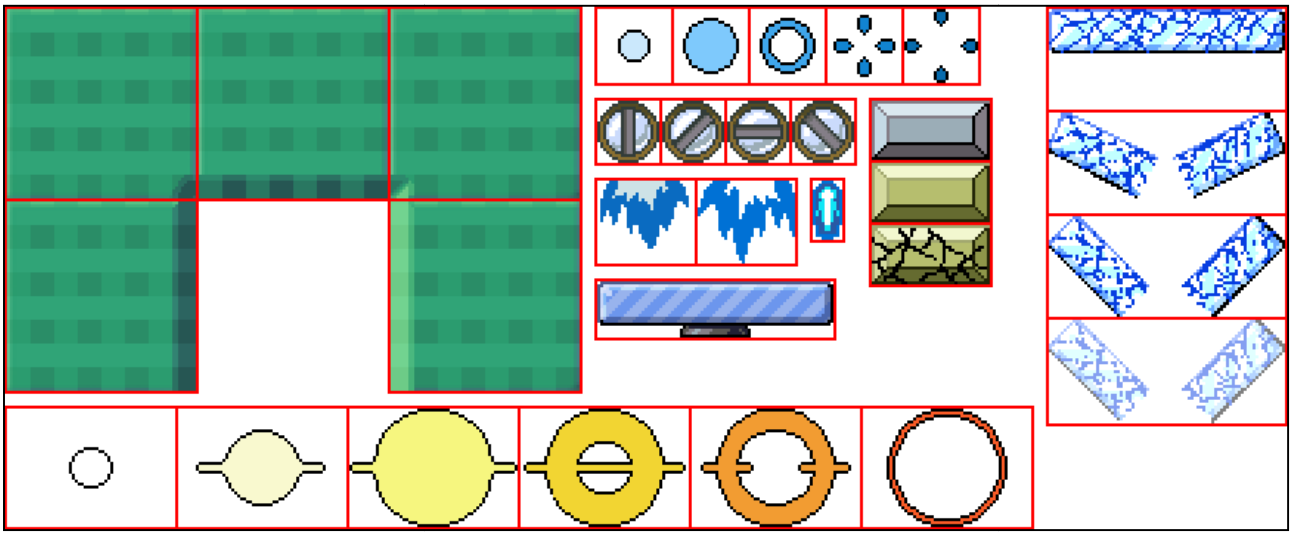


Los métodos usados por los canales de textura para leer y ampliar las imágenes (mapeo de pixels, copias internas, conexiones hardware, etc.) dependen de la implementación.

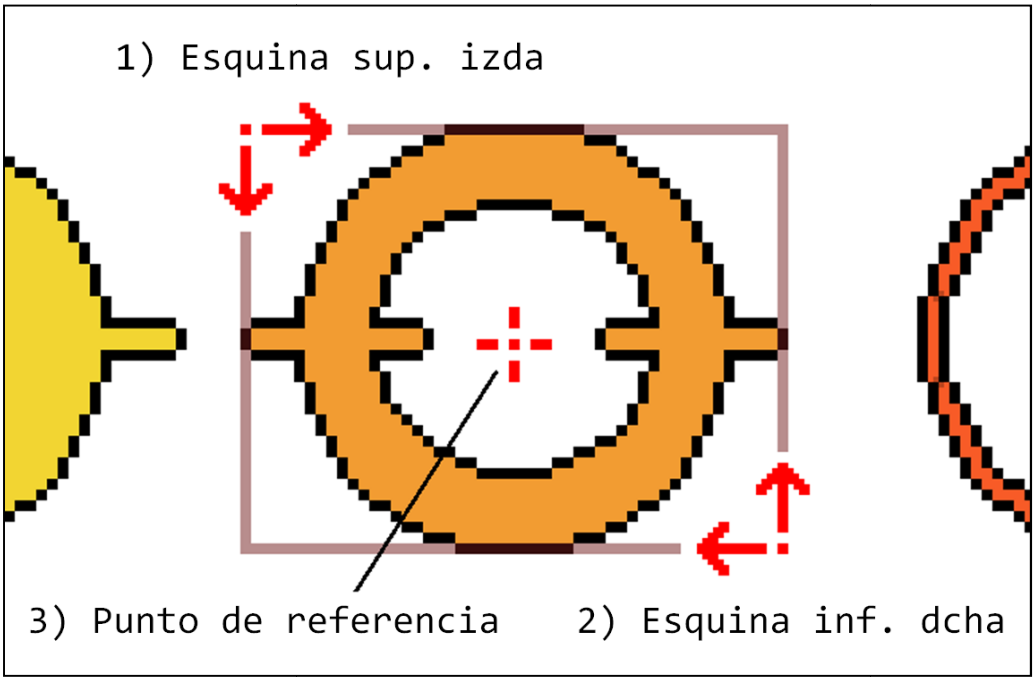
3.5 Regiones de texturas

La GPU no trabaja directamente con texturas, ya que éstas son más grandes que la pantalla. En vez de eso se puede definir un conjunto de regiones delimitadas dentro de cada textura para que las funciones de dibujo de la GPU operen con ellas.

Definir regiones para cada textura permite a los programas de Vircon32 usar las texturas de forma más eficiente agrupando varias imágenes como parte de una única textura. Se puede apreciar cómo funciona esta estrategia en el siguiente ejemplo:



Una región de textura es una parte rectangular de esa textura, definida por 3 pixels como se ve en la siguiente imagen. Los pixels 1 y 2 definen los límites de la región (ambos están incluidos en ella) mientras que el pixel 3, llamado el foco, se usa como referencia para colocar la región en la pantalla al dibujar. El foco puede definirse fuera de la región.



3.6 Elementos seleccionados

La GPU organiza sus elementos gráficos (texturas y regiones) en conjuntos, pero normalmente sólo opera con un elemento de cada conjunto en un momento dado.

Textura seleccionada

Todos los canales de textura asignados forman un rango continuo de IDs usables como se ve en esta imagen. Aún así, sólo una de las texturas disponibles estará activa en cada momento. Para decidir qué textura usar al dibujar o hacer cualquier otra función, la GPU siempre considera uno de esos IDs "seleccionado". La textura seleccionada por defecto es la de la BIOS, ya que es la única que se garantiza que siempre existe.

Región seleccionada

Para cada canal de textura asignado hay 4096 regiones configurables, accesibles a través de los IDs de región de 0 a 4095. Debe tenerse en cuenta que todas ellas siempre existirán y podrán usarse, aún si el programa no ha definido sus valores. Igual que con las texturas, la GPU siempre considera uno de los IDs de región como "seleccionado", para determinar qué región usan las funciones de la GPU. Este parámetro, sin embargo, es global: no hay una región seleccionada diferente para cada textura.

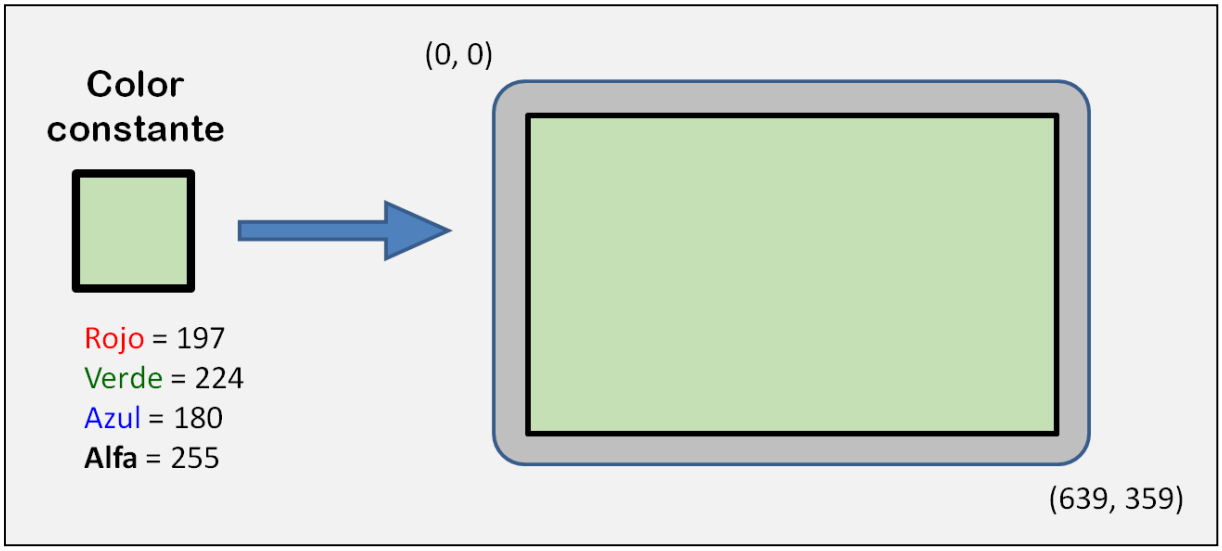
4 Funciones de dibujo

Como se mencionó en la sección de introducción, sólo hay 2 maneras en que la GPU puede dibujar en la pantalla: borrar la pantalla o dibujar una región de textura. Esta sección dará una descripción básica de cómo se realizan estas funciones.

Sin embargo, el detalle completo de cómo se realizan las funciones gráficas se cubrirá más adelante, en la sección que describe la ejecución de comandos. Además, aunque es posible aplicar ciertos efectos modificadores, aquí sólo se describe la versión sin modificar.

4.1 Borrar la pantalla

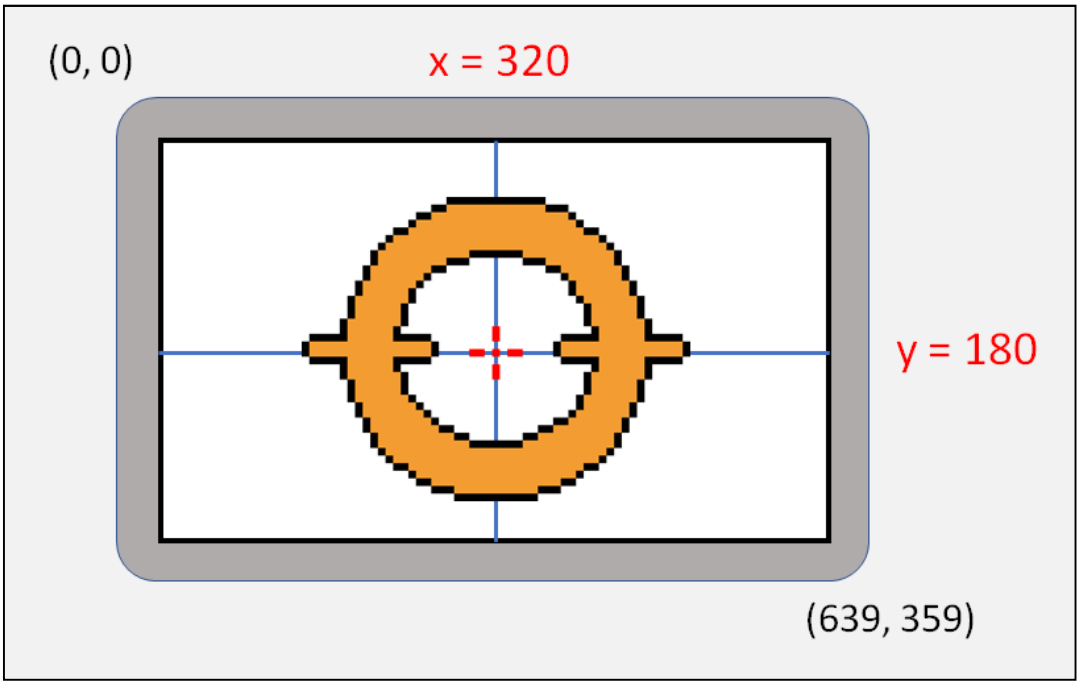
Un borrado de pantalla aplicará un único color (el color de borrado) sobre todos los pixels del buffer de dibujo. Borrar la pantalla es la función más sencilla: funciona sin acceder a ninguna textura y no puede modificarse mediante efectos gráficos.



4.2 Dibujar regiones de texturas

La GPU puede dibujar cualquiera de las regiones de textura disponibles en el buffer de dibujo. Para determinar dónde colocar esa región usa una posición de dibujo, dada por sus coordenadas en pixels. La región se dibuja de forma que su foco o punto de referencia se sitúe en dicha posición de dibujo.

Por ejemplo, si la posición de dibujo es el centro de la pantalla (pixel 320,180), el punto de referencia de la región de muestra que definimos anteriormente se hará coincidir con estas coordenadas como en el siguiente ejemplo.

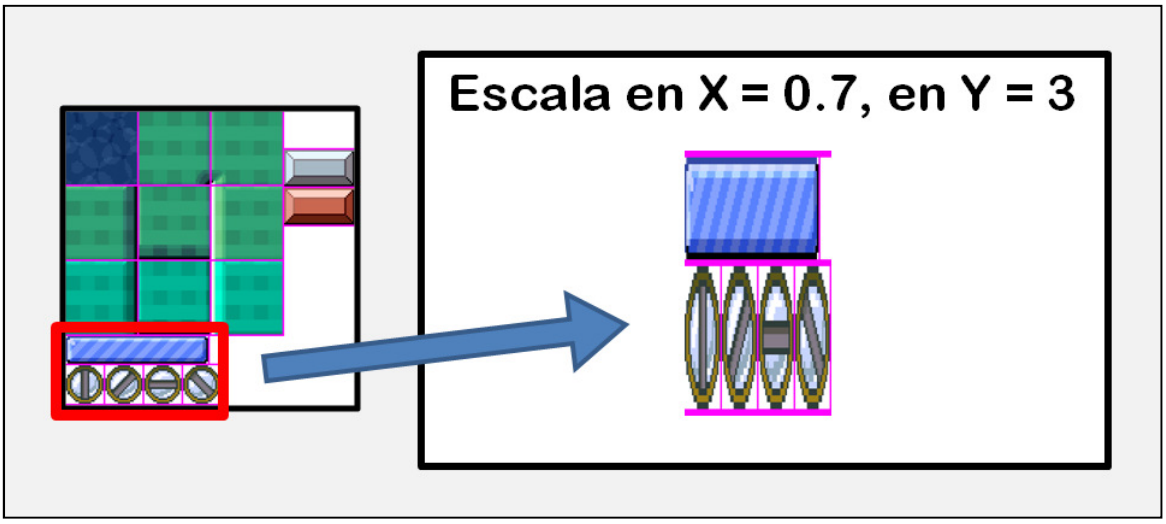


5 Efectos gráficos

Para tener mayor flexibilidad y características gráficas más avanzadas, se pueden aplicar distintos efectos a las funciones de dibujo de la GPU. Estos efectos, cuando se activan, modifican las funciones de dibujo y, por tanto, cambian el resultado en pantalla.

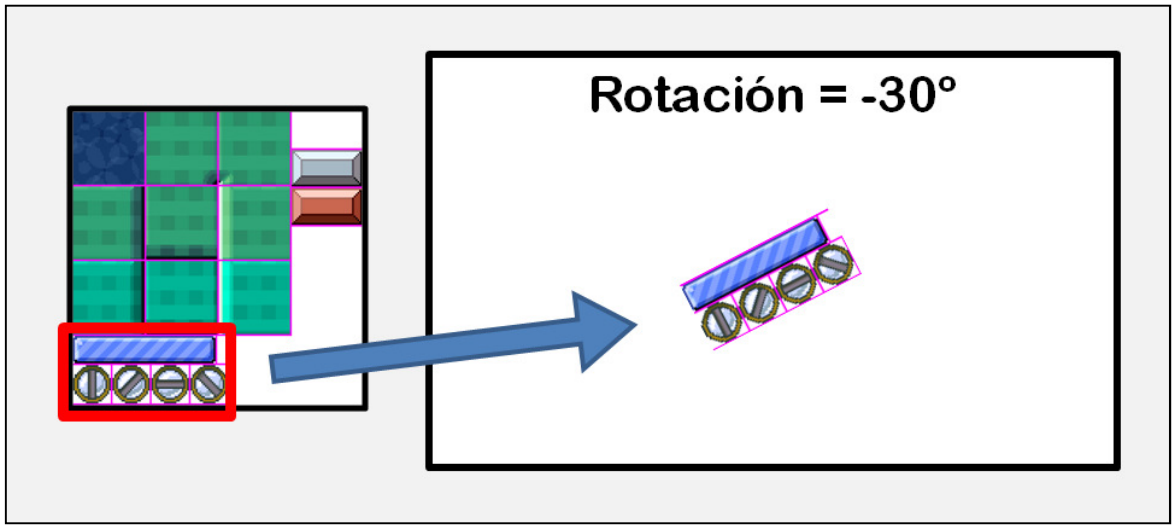
5.1 Efecto de escalado

Al dibujar una región de textura se puede indicar a la GPU que aplique factores de escala según los ejes X e Y de la región. Como resultado el rectángulo de salida se escala en consecuencia, preservando la posición del foco. Un ejemplo de esto podría ser el siguiente:



5.2 Efecto de rotación

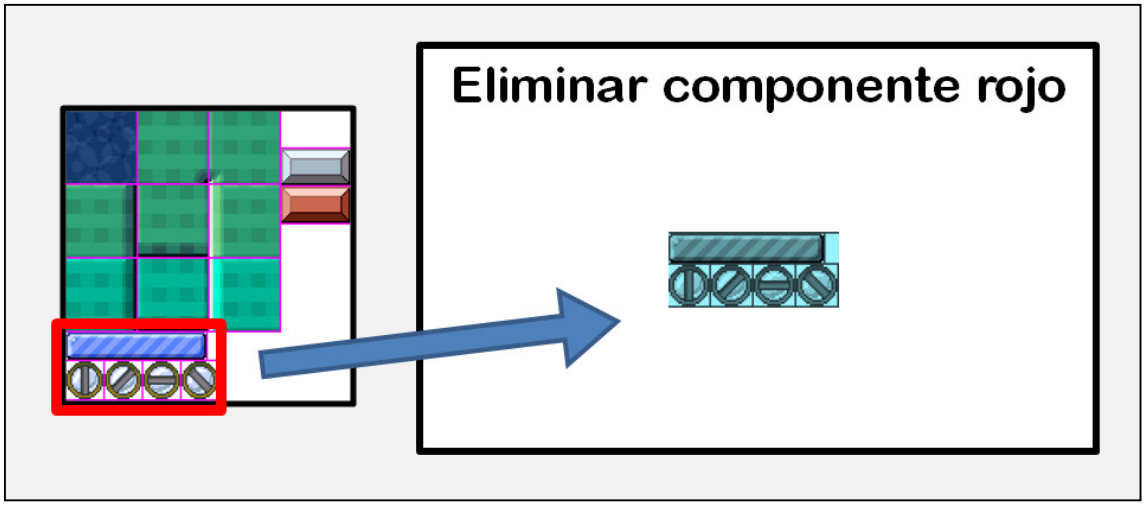
Otro efecto opcional disponible al dibujar una región de textura es aplicar un ángulo de rotación. La región dibujada se girará con respecto a su foco, lo que de nuevo conserva su posición en la pantalla. Un ejemplo de región rotada podría ser el siguiente:



5.3 Efecto de multiplicación del color

Este efecto se aplica siempre que se dibujan regiones. Los componentes de color de cada pixel que se va a dibujar se multiplican primero por los de un único color dado (el color de multiplicado). El efecto del color multiplicado es el de un filtro: modulará los 4 componentes de color de cada pixel de la región antes de que se dibujen en la pantalla. Este efecto es el mismo que cuando se usa la función glColor en OpenGL.

Como ejemplo de este efecto, si usamos un color de multiplicado de (0, 255, 255, 255), esto eliminaría el componente rojo de la región dibujada conservando los otros componentes. El efecto visual mostrado en pantalla sería el siguiente:



Para cada pixel de la región se modifica el color dibujado de acuerdo a estas fórmulas:

```
R Dibujado = (R Pixel * R Multiplicado) / 255
G Dibujado = (G Pixel * G Multiplicado) / 255
B Dibujado = (B Pixel * B Multiplicado) / 255
A Dibujado = (A Pixel * A Multiplicado) / 255
```

El efecto de modulación para cada componente podría verse mejor si estas fórmulas se escribieran como una proporción: X Pixel * (X Multiplicado / 255). Sin embargo, para algunas implementaciones, este orden de operaciones puede dar un resultado incorrecto.

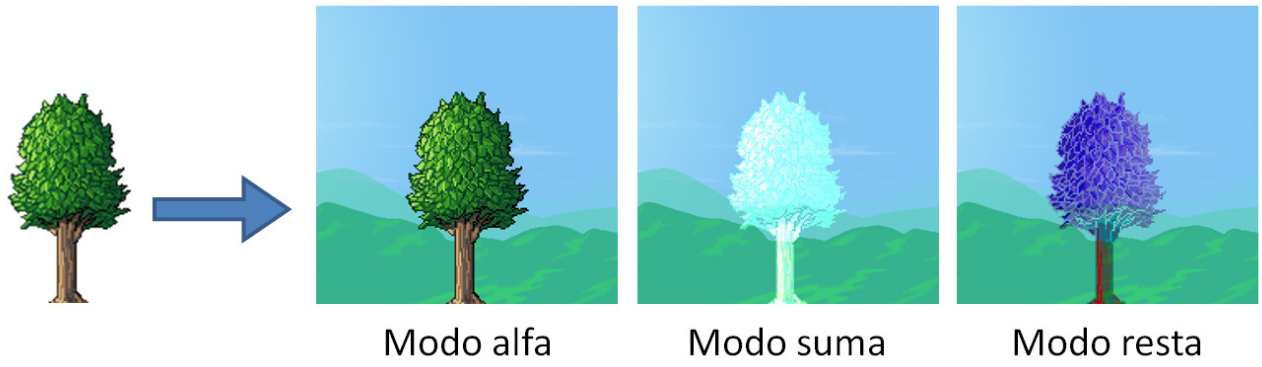
Un color de multiplicado blanco total no tiene efecto sobre los colores dibujados. Como este es el valor por defecto, en el arranque este efecto puede considerarse “desactivado”.

6 Mezclado de colores

La mezcla de colores es el proceso usado para dibujar un nuevo color en un pixel del buffer de dibujo. Es necesario definir el proceso porque los gráficos de Vircon32 soportan transparencias básicas y efectos de iluminación. En un sistema gráfico así, dibujar un

nuevo color no siempre significa que el color dibujado sustituirá al color anterior del pixel. Además también puede depender del color anterior del pixel, de diferentes formas.

La GPU admite 3 modos diferentes de mezcla de colores. En cada momento dado uno de los 3 modos estará activo, y cualquier función gráfica que se ejecute usará ese modo para dibujar en el buffer. Esta imagen puede dar una idea rápida de cómo son los 3 modos:



6.1 Mezclado alfa

Este es el modo de mezcla por defecto. Sólo dibuja los colores tal y como son, teniendo en cuenta el canal alfa para determinar el nivel de transparencia sobre el fondo. En su forma más básica (opacidad total, sin transparencias), su efecto sería simplemente: Color Buffer = Color Dibujado. Cuando tenemos en cuenta el alfa el efecto se modula de esta forma:

```
R Buffer = (R Dibuj. * A Dibuj. + R Buffer * (255 – A Dibuj.)) / 255
G Buffer = (G Dibuj. * A Dibuj. + G Buffer * (255 – A Dibuj.)) / 255
B Buffer = (B Dibuj. * A Dibuj. + B Buffer * (255 – A Dibuj.)) / 255
```

El resultado ya estará dentro del rango [0-255], por lo que no se requiere recorte. Cualquier color dibujado con transparencia total no produce cambios.

6.2 Mezclado aditivo

Este modo se usa para aclarar los colores y producir efectos de luz. En su forma más básica (opacidad total), su efecto sería sólo: Color Buffer += Color Dibujado, y luego recortar a [0-255]. Al añadir un canal alfa el efecto se modula de esta forma:

```
R Buffer =  R Buffer + (R Dibujado * A Dibujado) / 255
G Buffer =  G Buffer + (G Dibujado * A Dibujado) / 255
B Buffer =  B Buffer + (B Dibujado * A Dibujado) / 255
```

Los 3 componentes RGB resultantes se limitan a un máximo de 255. Cuando el color dibujado es negro o totalmente transparente, no produce ningún cambio. Este modo de

mezclado debería ser análogo a las capas en modos "add" ó "linear dodge" en muchos programas de dibujo.

6.3 Mezclado sustractivo

Este modo se usa para oscurecer los colores y producir efectos de sombra. En su forma más básica (opacidad total), su efecto sería sólo: Color Buffer –= Color Dibujado, y luego recortar a [0-255]. Al añadir un canal alfa el efecto se modula de esta forma:

```
R Buffer =  R Buffer - (R Dibujado * A Dibujado) / 255
G Buffer =  G Buffer - (G Dibujado * A Dibujado) / 255
B Buffer =  B Buffer - (B Dibujado * A Dibujado) / 255
```

Los 3 componentes RGB resultantes se limitan a un mínimo de 0. Cuando el color dibujado es negro o transparente, no hay ningún cambio. Este modo de mezclado debería ser análogo a las capas en modos “subtract” ó “difference” muchos programas de dibujo.

7 Rendimiento de la GPU

La GPU no estaría totalmente definida sin establecer sus límites de rendimiento. Para esta GPU el rendimiento se basa en una estimación aproximada de la cantidad de pixels dibujados. En cada frame la GPU puede dibujar 9 veces la pantalla completa. Es decir, la misma cantidad de pixels que una pantalla completa de 1920 x 1080 pixels.

7.1 Cálculo de los pixels dibujados

Lo primero que debe hacer la GPU al recibir una petición válida de una operación de dibujo es calcular una estimación simplificada del número de pixels que se dibujarán.

Para un borrado de pantalla la cantidad de pixels dibujados es siempre 1 pantalla completa = 640 x 360 pixels. Al dibujar regiones, el cálculo sigue estos pasos:

1) Calcular el "ancho efectivo” así: Primero toma el ancho de la región en pixels de textura. Si se aplica escalado, se multiplica por el factor de escala en X. Se toma el valor absoluto y, si es mayor que el ancho de la pantalla, se limita a 640 pixels.

2) Calcular el "alto efectivo" así: Primero toma el alto de la región en pixels de textura. Si se aplica escalado, se multiplica por el factor de escala en Y. Se toma el valor absoluto y, si es mayor que el alto de la pantalla, se limita a 360 pixels.

3) Finalmente calcular los “pixels dibujados” como "ancho efectivo" x "alto efectivo".

Nótese que este cálculo ignora la posición de dibujado y el ángulo de rotación. Esto hace que las partes de una región que queden fuera de pantalla también se cuenten como pixels dibujados. Pero esta imprecisión simplifica mucho el cálculo en algunos casos.

7.2 Factores de rendimiento para las distintas operaciones

No todas las operaciones cuestan lo mismo a la GPU. Para modelar esto, según la operación realizada, el número de pixels dibujados se modificará con estos factores:

- Al borrar la pantalla: –50%
- Al dibujar regiones aplicando escalado: +15%
- Al dibujar regiones aplicando rotación: +25%

(si se aplica tanto escalado como rotación, el factor combinado es +40%)

Nótese que las operaciones de procesado de color, como aplicar el color de multiplicado y el mezclado de colores, no afectan al rendimiento y no tienen factores asociados.

7.3 Comprobar el consumo de un comando

Tras determinar el número de pixels a dibujar para la petición de comando recibida, la GPU comparará esa cantidad con sus pixels restantes actuales.

- Si hay suficientes pixels restantes para la operación de dibujo en este frame, entonces ejecutará el comando y reducirá los pixels restantes según lo calculado.

- Si, por el contrario, los pixels restantes actuales no son suficientes para la operación, la GPU dejará sus pixels restantes en –1 y no ejecutará el comando. Cualquier otra petición recibida dentro del frame actual también será ignorada.

Debido a este último punto, las implementaciones pueden simplificar este proceso y rechazar automáticamente las operaciones sin ningún cálculo cuando el número de pixels restantes sea menor o igual a 0.

La capacidad de dibujo de la GPU no puede transferirse entre frames. Un nuevo frame restablecerá la misma capacidad de pixels independientemente de los pixels restantes en el frame anterior.

8 Variables internas

La GPU dispone de un conjunto de variables que guardan distintos aspectos de su estado interno. Cada una de estas variables se almacena como un valor de 32 bits, y todas ellas se interpretan con los mismos formatos (entero, float, etc.) descritos en la parte 2 de la especificación. A continuación las enumeramos y detallamos, organizadas en secciones.

8.1 Variables para el control de la GPU

Pixels restantes	**Valor inicial:** 2073600 (= 9 * 640 * 360)
Formato: Entero	**Rango válido:** De -1 a 2073600

Almacena la capacidad de dibujo restante de la GPU para este frame. Cuando una petición de dibujo no se puede realizar (es decir, los pixels restantes no son suficientes para ese comando), la GPU se bloquea hasta que comience el siguiente frame y ya no realizará ningún comando en el frame actual. Se utiliza un valor de –1 para indicar que se ha agotado la capacidad de dibujo para este frame.

8.2 Parámetros espaciales de dibujo

X punto de dibujo	**Valor inicial:** 0
Formato: Entero	**Rango válido:** De -1000 a 1639

Marca la coordenada X, en pixels, donde se situará en pantalla el foco de la siguiente región dibujada. Ésta puede quedar fuera del ancho de la pantalla, lo que puede provocar que la región dibujada sea no visible parcial o totalmente.

Y punto de dibujo	**Valor inicial:** 0
Formato: Entero	**Rango válido:** De -1000 a 1359

Marca la coordenada Y, en pixels, donde se situará en pantalla el foco de la siguiente región dibujada. Ésta puede quedar fuera del alto de la pantalla, lo que puede provocar que la región dibujada sea no visible parcial o totalmente.

Escala de dibujo en X	**Valor inicial:** 1.0
Formato: Float	**Rango válido:** De -1024.0 a 1024.0

Este parámetro de dibujo sólo se aplica con comandos que permitan el escalado. Cuando se usa, las regiones dibujadas se escalarán según la dimensión X de la textura con este factor. También puede ser negativo, para un efecto espejo según X.

Si se dibuja con una escala de reducción y el ancho resultante es menor de 1 pixel, el resultado se considera dependiente de la implementación: podría comprimir la región en una línea vertical de 1 pixel, o no dibujar nada en absoluto.

Escala de dibujo en Y	**Valor inicial:** 1.0
Formato: Float	**Rango válido:** De -1024.0 a 1024.0

Este parámetro de dibujo sólo se aplica con comandos que permitan el escalado. Cuando se usa, las regiones dibujadas se escalarán según la dimensión Y de la textura con este factor. También puede ser negativo, para un efecto espejo según Y.

Si se dibuja con una escala de reducción y el alto resultante es menor de 1 pixel, el resultado se considera dependiente de la implementación: podría comprimir la región en una línea horizontal de 1 pixel, o no dibujar nada en absoluto.

Ángulo de dibujo	**Valor inicial:** 0.0
Formato: Float	**Rango válido:** De -1024.0 a 1024.0

Este parámetro de dibujo sólo se aplica con comandos que permitan la rotación. Es el ángulo de rotación que se aplica a las regiones dibujadas. El valor se interpreta en radianes (1 círculo completo = 2*pi radianes = 360 grados). Para un ángulo de 0 no hay rotación, y para valores positivos las regiones se rotan según las agujas del reloj.

8.3 Variables de procesado de color

Color de borrado	**Valor inicial:** R = 0, G = 0, B = 0, A = 255
Formato: Color GPU	**Rango válido:** Todo el rango RGBA

El color de borrado actual es el que se aplicará siempre que se ejecute un comando de borrar pantalla. Soporta la transparencia modulando el componente alfa.

Color de multiplicado	**Valor inicial:** R = 255, G = 255, B = 255, A = 255
Formato: Color GPU	**Rango válido:** Todo el rango RGBA

Este color controla el efecto de multiplicado de color que se aplica siempre en cualquier comando de dibujar regiones. El valor inicial es el color de multiplicado neutro (sin efecto).

Modo de mezclado activo	**Valor inicial:** 20h (mezclado alfa)
Formato: Entero	**Rango válido:** Sólo los valores listados

Este valor se interpreta como el modo de mezclado de color actualmente activo. Éste controla cómo se dibujan los colores en todas las funciones de dibujo. El funcionamiento de cada modo se puede consultar en la sección de mezclado de colores. Los valores posibles son los siguientes:

20h: Mezclado alfa
21h: Mezclado aditivo
22h: Mezclado sustractivo

8.4 Elementos seleccionados

Textura seleccionada	**Valor inicial:** -1
Formato: Entero	**Rango válido:** De -1 a 255 (*)

Este valor es el ID numérico de la textura de GPU seleccionada actualmente. La textura seleccionada es la que se usará en todos los comandos de dibujo de regiones. También es la textura cuyas regiones se verán afectadas si hay cambios en la configuración de región.

(*) El límite superior viene dado por el cartucho conectado en ese momento. Si no hay ningún cartucho, la textura de la BIOS (ID = -1) es la única seleccionable.

Región seleccionada	**Valor inicial:** 0
Formato: Entero	**Rango válido:** De 0 a 4095

Este valor es el ID numérico de la región actualmente seleccionada, dentro de la textura seleccionada. Cambiar la textura seleccionada no cambia el ID de región seleccionada: es un parámetro global (no hay una región seleccionada para cada textura). La región seleccionada (de la textura seleccionada) es la que se usa en todos los comandos de dibujo de regiones. También es la región afectada por cambios en la configuración de región.

8.5 Configuración de cada región de textura

Las variables listadas aquí son especiales. La GPU guarda una copia de cada una de estas variables por cada región de textura posible. Esto significa que puede haber hasta (256+1) texturas * 4096 regiones/textura. La implementación puede decidir si todas las copias se guardan siempre, dejando inaccesibles las de canales de textura no usados, o si sólo se crean las necesarias cada vez que se inserta un nuevo cartucho.

Juntas, cada conjunto de estas variables describe la configuración actual para una única región de GPU. En la sección de regiones de textura (capítulo 3) se puede consultar cómo se definen las regiones. Debe destacarse que al definir la extensión de una región la GPU acepta que los mínimos y máximos estén invertidos, causando el volteo de la imagen.

Como siempre hay accesible un sólo conjunto de estas variables, se accede a cada variable de esta sección por puertos E/S a través de un “puntero”. Si la textura o región seleccionadas cambian, los puertos se redirigen a la copia de variables para esa región.

X mínima región	**Valor inicial:** 0
Formato: Entero	**Rango válido:** De 0 a 1023

Representa la coordenada X más a la izquierda de la textura que se considerará parte de la región. Dada en pixels, en coordenadas de textura.

Y mínima región	**Valor inicial:** 0
Formato: Entero	**Rango válido:** De 0 a 1023

Representa la coordenada Y más hacia arriba de la textura que se considerará parte de la región. Dada en pixels, en coordenadas de textura.

X máxima región	**Valor inicial:** 0
Formato: Entero	**Rango válido:** De 0 a 1023

Representa la coordenada X más a la derecha de la textura que se considerará parte de la región. Dada en pixels, en coordenadas de textura.

Y máxima región	**Valor inicial:** 0
Formato: Entero	**Rango válido:** De 0 a 1023

Representa la coordenada Y más hacia abajo de la textura que se considerará parte de la región. Dada en pixels, en coordenadas de textura.

X foco de región	**Valor inicial:** 0
Formato: Entero	**Rango válido:** De -1024 a 2047

Es la coordenada X, en coordenadas de textura, que se tomará como referencia al dibujar la región. La GPU calculará la ubicación de la región de forma que el foco se dibuje en el punto de dibujo. Se permite que el foco esté fuera de los límites de la textura, dentro de los límites del rango válido.

Y foco de región	**Valor inicial:** 0
Formato: Entero	**Rango válido:** De -1024 a 2047

Es la coordenada Y, en coordenadas de textura, que se tomará como referencia al dibujar la región. La GPU calculará la ubicación de la región de forma que el foco se dibuje en el punto de dibujo. Se permite que el foco esté fuera de los límites de la textura, dentro de los límites del rango válido.

9 Puertos de control

Esta sección detalla el conjunto de puertos de control que la GPU expone a través de su conexión al bus de control de la CPU como dispositivo esclavo. Todos los puertos expuestos junto con sus propiedades básicas se enumeran en la siguiente tabla:

Lista de puertos de control expuestos			
Dirección externa	Dirección interna	Nombre del puerto	Acceso L/E
200h	00h	Comando	Sólo Escritura
201h	01h	Pixels Restantes	Sólo Lectura
202h	02h	Color de Borrado	Lectura y Escritura
203h	03h	Color de Multiplicado	Lectura y Escritura
204h	04h	Mezclado Activo	Lectura y Escritura
205h	05h	Textura Seleccionada	Lectura y Escritura
206h	06h	Región Seleccionada	Lectura y Escritura
207h	07h	X Punto de Dibujo	Lectura y Escritura
208h	08h	Y Punto de Dibujo	Lectura y Escritura
209h	09h	Escala de Dibujo en X	Lectura y Escritura
20Ah	0Ah	Escala de Dibujo en Y	Lectura y Escritura
20Bh	0Bh	Ángulo de Dibujo	Lectura y Escritura
20Ch	0Ch	X Mínima Región	Lectura y Escritura
20Dh	0Dh	Y Mínima Región	Lectura y Escritura
20Eh	0Eh	X Máxima Región	Lectura y Escritura
20Fh	0Fh	Y Máxima Región	Lectura y Escritura
210h	10h	X Foco Región	Lectura y Escritura
211h	11h	Y Foco Región	Lectura y Escritura

9.1 Acciones en peticiones de lectura/escritura de puertos

Los puertos de control de la GPU no son simples registros hardware. Los efectos causados por una petición de lectura/escritura a un puerto concreto suelen ser distintos de lectura o escritura de valores. Esta sección detalla cómo se comporta cada puerto de la GPU.

Obsérvese que, además de las acciones realizadas, será necesario dar una respuesta de éxito/fallo a la petición, como parte de la comunicación del bus de control. Si no se indica lo contrario, siempre se asumirá que la respuesta es de éxito. Cuando la respuesta sea de fallo, la GPU no realizará más acciones y la CPU activará un error hardware.

Puerto “Comando”

En peticiones de lectura:
Como este puerto es de sólo escritura, se dará una respuesta de fallo al bus de control.

En peticiones de escritura:
La GPU realizará el proceso de ejecución de comando detallado en la sección 10. En todos los casos se dará respuesta de éxito a la petición, aún si el comando no se ejecuta.

Puerto "Pixels Restantes"

En peticiones de lectura:
La GPU proveerá el valor actual de la variable interna "Pixels restantes".

En peticiones de escritura:
Como este puerto es de sólo lectura, se dará una respuesta de fallo al bus de control.

Puerto "Color de Borrado"

En peticiones de lectura:
La GPU proveerá el valor actual de la variable interna "Color de borrado".

En peticiones de escritura:
La GPU sobrescribirá la variable interna "Color de borrado" con el valor recibido. Esto hará que las siguientes operaciones de borrado de pantalla apliquen el nuevo color. Obsérvese que escribir esta variable no provoca un borrado de pantalla.

Puerto "Color de Multiplicado"

En peticiones de lectura:
La GPU proveerá el valor actual de la variable interna "Color de multiplicado".

En peticiones de escritura:
La GPU sobrescribirá la variable interna "Color de multiplicado" con el valor recibido. Esto hará que las siguientes operaciones de dibujo de regiones apliquen inmediatamente el nuevo color de multiplicado.

Puerto "Mezclado Activo"

En peticiones de lectura:
La GPU proveerá el valor actual de la variable interna "Modo de mezclado activo".

En peticiones de escritura:

La GPU comprobará si el valor recibido corresponde a un modo de mezclado válido. Si no lo es, la petición será ignorada. Para valores válidos, la GPU sobrescribirá la variable interna "Modo de mezclado activo" con el valor recibido. Esto hará que las siguientes operaciones de dibujo apliquen inmediatamente el nuevo modo de mezclado.

Puerto “Textura Seleccionada”

En peticiones de lectura:

La GPU proveerá el valor actual de la variable interna “Textura seleccionada”.

En peticiones de escritura:

La GPU comprobará si el valor recibido corresponde a un ID de textura actualmente válido. Si no lo es la petición será ignorada. Para valores válidos, la GPU sobrescribirá la variable interna “Textura seleccionada” con el valor recibido. Después redirigirá todos los puertos de configuración de región para que apunten al conjunto de variables de la nueva región seleccionada (es decir, el mismo ID de región, pero para la nueva textura).

Puerto “Región Seleccionada”

En peticiones de lectura:

La GPU proveerá el valor actual de la variable interna “Región seleccionada”.

En peticiones de escritura:

La GPU comprobará si el valor recibido corresponde a un ID de región válido. Si no lo es la petición será ignorada. Para valores válidos, la GPU sobrescribirá la variable interna “Región seleccionada” con el valor recibido. Después, redirigirá todos los puertos de configuración de región para que apunten al conjunto de variables de la nueva región seleccionada (es decir, el nuevo ID de región, pero de la misma textura).

Puerto “X Punto de Dibujo”

En peticiones de lectura:

La GPU proveerá el valor actual de la variable interna “X punto de dibujo”.

En peticiones de escritura:

La GPU comprobará si el valor recibido está fuera del rango de la variable interna "X Punto de dibujo" y, en tal caso, lo ajustará a los límites del rango. El valor resultante sobrescribirá la variable interna “X punto de dibujo”. Esto hará que las siguientes operaciones de dibujo apliquen inmediatamente la nueva posición de dibujo.

Puerto “Y Punto de Dibujo”

En peticiones de lectura:
La GPU proveerá el valor actual de la variable interna “Y punto de dibujo”.

En peticiones de escritura:
La GPU comprobará si el valor recibido está fuera del rango de la variable interna “Y punto de dibujo” y, en tal caso, lo ajustará a los límites del rango. El valor resultante sobrescribirá la variable interna “Y punto de dibujo”. Esto hará que las siguientes operaciones de dibujo apliquen inmediatamente la nueva posición de dibujo.

Puerto “Escala de Dibujo en X”

En peticiones de lectura:
La GPU proveerá el valor actual de la variable interna “Escala de dibujo en X”.

En peticiones de escritura:
La GPU comprobará si el valor recibido está fuera del rango de la variable interna “Escala de dibujo en X” y, en tal caso, lo ajustará a los límites del rango. El valor resultante sobrescribirá la variable interna “Escala de dibujo en X”. Esto hará que las siguientes operaciones de dibujo apliquen inmediatamente la nueva escala de dibujo en X.

Puerto “Escala de Dibujo en Y”

En peticiones de lectura:
La GPU proveerá el valor actual de la variable interna “Escala de dibujo en Y”.

En peticiones de escritura:
La GPU comprobará si el valor recibido está fuera del rango de la variable interna “"Escala de dibujo en Y” y, en tal caso, lo ajustará a los límites del rango. El valor resultante sobrescribirá la variable interna “Escala de dibujo en Y”. Esto hará que las siguientes operaciones de dibujo apliquen inmediatamente la nueva escala de dibujo en Y.

Puerto “Ángulo de Dibujo”

En peticiones de lectura:
La GPU proveerá el valor actual de la variable interna “Ángulo de dibujo”.

En peticiones de escritura:

La GPU comprobará si el valor recibido está fuera del rango de la variable interna "Ángulo de dibujo" y, en tal caso, lo ajustará a los límites del rango. El valor resultante sobrescribirá la variable interna "Ángulo de dibujo". Esto hará que las siguientes operaciones de dibujo con rotación activada apliquen el nuevo ángulo de dibujo.

Puerto "X Mínima Región"

En peticiones de lectura:

La GPU proveerá el valor actual de la variable interna "X mínima región" asociada al ID de región actualmente seleccionado, para el ID de textura actualmente seleccionado.

En peticiones de escritura:

La GPU comprobará si el valor recibido está fuera del rango de la variable interna "X mínima región" y, en tal caso, lo ajustará a los límites del rango. El valor resultante sobrescribirá la variable interna "X mínima región" asociada al ID de región actualmente seleccionado, para el ID de textura actual. Esto hará que las siguientes operaciones de dibujo de regiones apliquen inmediatamente el nuevo límite izquierdo para esa región.

Puerto "Y Mínima Región"

En peticiones de lectura:

La GPU proveerá el valor actual de la variable interna "Y mínima región" asociada al ID de región actualmente seleccionado, para el ID de textura actualmente seleccionado.

En peticiones de escritura:

La GPU comprobará si el valor recibido está fuera del rango de la variable interna "Y mínima región" y, en tal caso, lo ajustará a los límites del rango. El valor resultante sobrescribirá la variable interna "Y mínima región" asociada al ID de región actualmente seleccionado, para el ID de textura actual. Esto hará que las siguientes operaciones de dibujo de regiones apliquen inmediatamente el nuevo límite superior para esa región.

Puerto "X Máxima Región"

En peticiones de lectura:

La GPU proveerá el valor actual de la variable interna "X máxima región" asociada al ID de región actualmente seleccionado, para el ID de textura actualmente seleccionado.

En peticiones de escritura:

La GPU comprobará si el valor recibido está fuera del rango de la variable interna "X máxima región" y, en tal caso, lo ajustará a los límites del rango. El valor resultante sobrescribirá la variable interna "X máxima región" asociada al ID de región actualmente seleccionado, para el ID de textura actual. Esto hará que las siguientes operaciones de dibujo de regiones apliquen inmediatamente el nuevo límite derecho para esa región.

Puerto "Y Máxima Región"

En peticiones de lectura:

La GPU proveerá el valor actual de la variable interna "Y máxima región" asociada al ID de región actualmente seleccionado, para el ID de textura actualmente seleccionado.

En peticiones de escritura:

La GPU comprobará si el valor recibido está fuera del rango de la variable interna "Y máxima región" y, en tal caso, se ajustará a los límites del rango. El valor resultante sobrescribirá la variable interna "Y máxima región" asociada al ID de región actualmente seleccionado, para el ID de textura actual. Esto hará que las siguientes operaciones de dibujo de regiones apliquen inmediatamente el nuevo límite inferior para esa región.

Puerto "X Foco de Región"

En peticiones de lectura:

La GPU proveerá el valor actual de la variable interna "X foco de región" asociada al ID de región actualmente seleccionado, para el ID de textura actualmente seleccionado.

En peticiones de escritura:

La GPU comprobará si el valor recibido está fuera del rango de la variable interna "X foco de región" y, en tal caso, se ajustará a los límites del rango. El valor resultante sobrescribirá la variable interna "X foco de región" asociada al ID de región actualmente seleccionado, para el ID de textura actual. Esto hará que las siguientes operaciones de dibujo de regiones apliquen inmediatamente la nueva posición del foco para esa región.

Puerto "Y Foco de Región"

En peticiones de lectura:

La GPU proveerá el valor actual de la variable interna "Y foco de región" asociada al ID de región actualmente seleccionado, para el ID de textura actualmente seleccionado.

En peticiones de escritura:
La GPU comprobará si el valor recibido está fuera del rango de la variable interna "Y foco de región" y, en tal caso, se ajustará a los límites del rango. El valor resultante sobrescribirá la variable interna "Y foco de región" asociada al ID de región actualmente seleccionado, para el ID de textura actual. Esto hará que las siguientes operaciones de dibujo de regiones apliquen inmediatamente la nueva posición del foco para esa región.

10 Ejecución de comandos

Los comandos de la GPU son operaciones de dibujo que pueden ser solicitadas por la CPU, enviando un valor al puerto "Comando" de la GPU. Esta sección describe el comportamiento de la GPU cuando recibe de estas una peticiones.

Hay 5 comandos diferentes que la GPU puede realizar. Esta tabla muestra sus valores numéricos, así como los efectos gráficos que se usan en cada comando.

Nombre del comando	Valor numérico	Escalado	Rotación	Color de multiplicado
Borrar Pantalla	10h	no	no	no
Dibujar Región	11h	no	no	sí
Dibujar Región Escalada	12h	sí	no	sí
Dibujar Región Rotada	13h	no	sí	sí
Dibujar Región Rotoescalada	14h	sí	sí	sí

10.1 Procesado común

El funcionamiento general descrito aquí es común a la ejecución de todos los comandos.

Como primer paso, la GPU comprobará si el valor de escritura pedido corresponde a uno de los comandos válidos enumerados anteriormente. Si no es así la petición se ignorará y se detendrá su procesado.

Después la GPU comprobará si los pixels restantes son suficientes para realizar el comando pedido, tal como se describe en la sección 7 (Rendimiento de la GPU).

Si el comando no se abortó por pixels restantes insuficientes, la GPU dibujará en el buffer de dibujo. Esta parte es específica de cada comando y se cubre en las siguientes subsecciones. Sin embargo, en todos los comandos, el mezclado de color se hará como se describe en la sección 6 para el valor actual de la variable "Modo de mezclado activo".

10.2 Comando "Borrar Pantalla"

La GPU dibujará en cada pixel del buffer de dibujo usando el color guardado actualmente en la variable interna "Color de borrado". Aparte del mezclado no se aplica ningún efecto.

10.3 Comando “Dibujar Región”

La GPU dibujará en el buffer de dibujo una parte rectangular de la textura actualmente seleccionada. Ese rectángulo está delimitado por las variables de configuración de región correspondientes al ID de región actualmente seleccionado, para esa textura.

La región se coloca de forma que el pixel marcado como foco de la región se dibuja en las coordenadas marcadas por la posición de dibujo actual de la GPU en el buffer de dibujo.

Al dibujar la región la GPU aplicará multiplicado de color, con el color de GPU guardado en la variable “Color de multiplicado”. El orden de las operaciones de color es este:

1) Hacer la multiplicación entre los pixels de la región y el color de multiplicado.
2) Hacer el mezclado del color resultante de (1) sobre el contenido del buffer de dibujo.

Este comando ignorará los valores actuales de las variables de escala y ángulo de dibujo. La región se dibuja siempre sin transformaciones espaciales.

10.4 Comando “Dibujar Región Escalada”

El procesado será el mismo que en el comando “Dibujar Región”, pero la región dibujada también se escala. Los factores de escala a lo largo de X e Y serán los valores de las variables "Escala de dibujo en X" y "Escala de dibujo en Y", respectivamente. Los factores de escala negativos a lo largo de X ó Y producen un efecto espejo en esa dimensión.

El escalado se hará de forma que, para cualquier factor de escala, el foco de la región se siga dibujando en la posición de dibujo de la GPU dentro del buffer de dibujo.

10.5 Comando “Dibujar Región Rotada”

El procesado será el mismo que en el comando “Dibujar Región”, pero la región dibujada también se rota. El ángulo de rotación será el valor de la variable “Ángulo de dibujo”. Se interpreta en radianes, y los ángulos positivos rotan en el sentido de las agujas del reloj.

La rotación se hará tomando como centro el foco de la región dibujada. Así el foco de la región siempre se dibuja en la posición de dibujo de la GPU dentro del buffer de dibujo.

10.6 Comando “Dibujar Región Rotoescalada”

El procesado será el mismo que en el comando “Dibujar Región”, pero la región dibujada también se escala y se rota. Estos efectos se aplican de la misma forma ya descrita para los comandos “Dibujar Región Escalada” y “Dibujar Región Rotada”.

Los efectos de escalado y rotación se combinarán garantizando que el foco de la región se siga dibujando en la posición de dibujo de la GPU dentro del buffer de dibujo. Además, la combinación de ambos efectos siempre debe realizar el escalado en relación con los ejes X

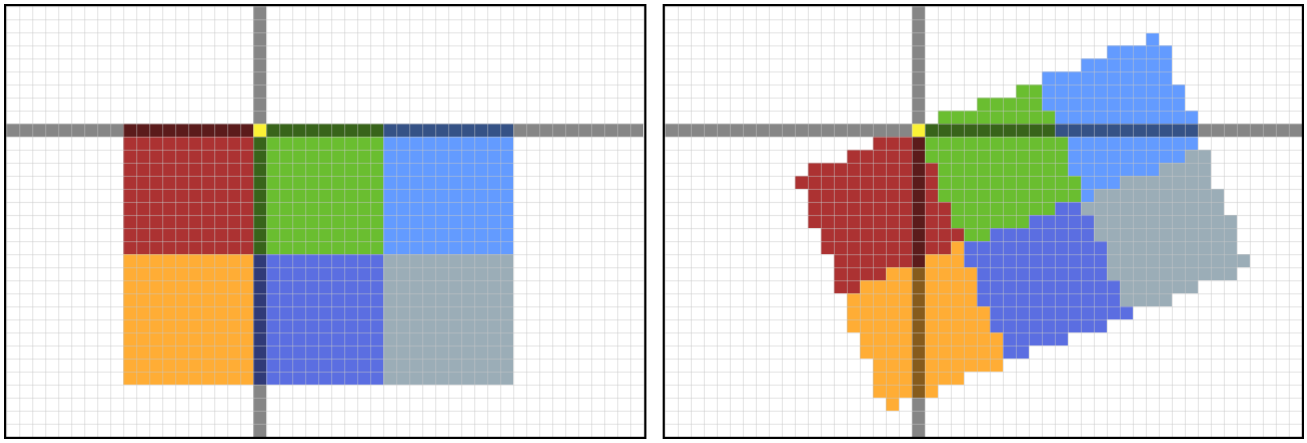
e Y de la textura y no en relación con las coordenadas de la pantalla. En otras palabras: el orden de las transformaciones debe ser primero el escalado y luego la rotación.

10.7 Notas sobre transformaciones 2D

Colocación de regiones para factores de escalado grandes

Al dibujar con escalas grandes, un solo pixel de textura puede ocupar varios pixels de la pantalla. En estos casos debe tenerse en cuenta que la referencia para colocar los pixels de la textura no es el centro del pixel sino su esquina superior izquierda. Esto significa que si escalamos un único pixel con los factores X = 640.0 e Y = 360.0, y lo dibujamos en la posición (0, 0) cubrirá exactamente la pantalla completa.

Como ejemplo, dibujaremos la región que se ve en la imagen (3 x 2 pixels) con una escala grande. El foco de esta región se sitúa en su pixel verde. Si usamos la posición de dibujo dada por la cruz de líneas oscurecidas, el resultado en pantalla será el siguiente:



Cuando la región se dibuja sin rotación (imagen izquierda), la esquina superior izquierda del pixel verde se solapa con la posición de dibujo. Si además activamos la rotación (imagen de la derecha), ese mismo pixel de la pantalla se usará como centro de rotación, por lo que la esquina original del pixel verde seguirá dibujándose en la posición de dibujo.

Métodos de interpolación

El método de interpolación de referencia en la GPU de Vircon32 es el del “vecino más próximo”. Aún así las implementaciones son libres de elegir otros métodos de interpolación para el escalado y la rotación de la imagen. Sin embargo, los algoritmos elegidos deben preservar los colores originales.

Esto significa que métodos con suavizado de color como la interpolación lineal o cúbica no serían compatibles con Vircon32. Pero algoritmos de pixel art como Scale2X o RotSprite serían aceptables, ya que están diseñados para conservar los colores de los pixels.

10.8 Tiempo de ejecución de los comandos

Esta especificación no menciona el tiempo que se tarda en ejecutar los comandos de dibujo. En su lugar, el sistema Vircon32 abstrae esto y trata todas las operaciones de dibujo como si siempre se terminaran instantáneamente. La consecuencia de esto es que la CPU podría pedir más comandos antes de que terminen los anteriores.

La razón de esta abstracción es que el sistema Vircon32 está diseñado para evitar tener que sincronizar operaciones entre componentes. En muchos casos esto es válido como aproximación: las GPUs modernas son mucho más rápidas que Vircon32. Aún así, en el mundo real algunas operaciones de dibujo pueden tardar varios ciclos de CPU. Esto hace que las implementaciones necesiten estrategias para gestionar las operaciones de dibujo.

Puede haber varios tipos de estrategias. En sistemas gráficos como OpenGL, las tareas se pueden realizar en segundo plano sin suponer un problema para el programa principal. En implementaciones de hardware, puede ser suficiente sólo con implementar una cola de comandos que se gestione en paralelo. Por otro lado, en una implementación de software se podría planificar toda la operación de la consola en torno a esto, y posponer los siguientes ciclos hasta que la última operación solicitada termine.

Aún así, por corrección gráfica, cualquier implementación debe garantizar que:

1) No se deja sin realizar ninguna operación de dibujo aceptada por la GPU.
2) Las operaciones de dibujo se realizan conservando su orden de petición.
3) Toda operación de dibujo se completa antes de que termine el frame actual.

11 Generación de la señal de video

La salida de video en esta GPU se rige por la temporización global de la consola y, por tanto, se basa en frames. A una frecuencia de 60 Hz, cada vez que se reciba la señal de nuevo frame, la GPU activará la generación y transmisión de un nuevo frame de video.

La GPU creará un frame de video agrupando toda la información de color necesaria para cada pixel del buffer de dibujo en ese momento. Luego la GPU transferirá esa información a la pantalla a través de la señal de salida de video. Si el formato/protocolo de la señal lo necesita, se añadirá información de temporizado o secuenciación para formar una transmisión válida de frames de video.

Las implementaciones son libres de elegir el formato de comunicación y los conectores físicos para las señales de audio y video. Pueden enviarse separadas o, como es habitual en pantallas actuales, usar un conector conjunto que envíe ambas a la misma pantalla.

Nótese que el contenido del buffer de dibujo no se borra ni se modifica cuando comienza un nuevo frame o como parte del proceso de salida de video. El contenido del buffer de dibujo es persistente y sólo se borra automáticamente en caso de reinicio o encendido.

12 Respuestas a señales de control

Como todos los componentes de la consola, cada vez que se produzca una señal de control, la GPU la recibirá y reaccionará para procesar ese evento. Para cada una de las señales de control, la GPU responderá realizando las siguientes acciones:

Señal de Reset:

- Si había operaciones de dibujo aún en curso, se abortan.
- Todas las variables internas de la GPU se establecen en sus valores iniciales. Esto incluye las variables de configuración para todas las regiones de todas las texturas.
- Cualquier efecto adicional asociado a esos cambios en las variables internas se aplica inmediatamente, como se indica en las peticiones de escritura en puertos de control.
- Todos los pixels del buffer de dibujo se restablecen a negro (R = 0, G = 0, B = 0).

Señal de Frame:

- Si había operaciones de dibujo aún en curso, se abortan.
- La GPU genera la salida para un nuevo frame de video, como indica la sección 11.
- La variable interna “Pixels restantes” se restablece a su valor inicial.

Señal de Ciclo:

- La GPU no necesita reaccionar a esta señal, a menos que se requiera por detalles concretos de la implementación.

Además de reaccionar a las señales de control, la GPU también tendrá que realizar el siguiente procesado en respuesta a eventos a nivel de la consola:

Cuando se conecta un nuevo cartucho:

- La GPU localizará las imágenes contenidas en la BIOS y, si está presente, el cartucho.
- La GPU realizará el proceso de conexión descrito en la sección 3 para asignar un canal de textura a cada imagen y acceder a la información de sus pixels.
- El límite superior de la variable “Textura seleccionada” se ajustará al ID del último canal de textura utilizado.

(Fin de la parte 4)