Spielekonsolenprogrammierung

Texturen

Übersicht

- Die Texturhardware der PSP
- Verwendung der LibGXT
  - Konvertieren von Texturen
  - Einbindung der Texturen

- Setzen der Texturen an den Fragmentshader
- Benutzung der Textur im Fragmentshader

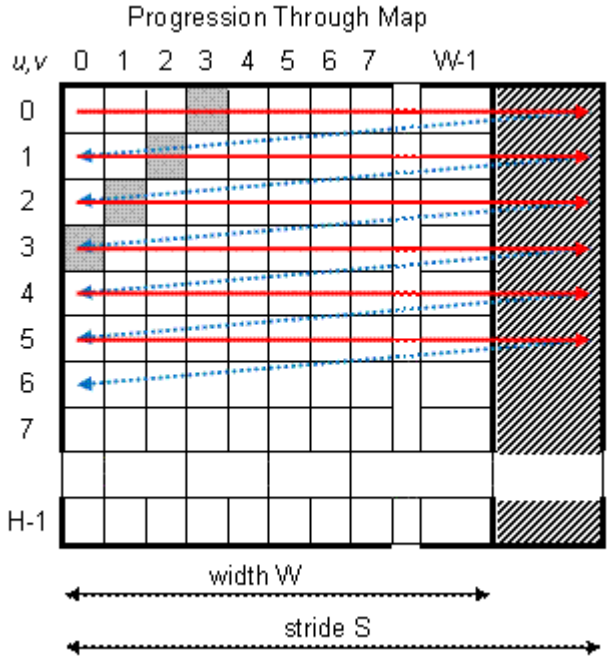
Texturhardware

- An Texturmöglichkeiten gibt es:
  - Standard RGBA Formate
  - Formate mit CLUT
  - Kompressionsformate UBC und PVR
  - Fließkommatexturen 16bit, 32bit
  - Cubemaps
  - Keine 3D Texturen!
- LibGXT deckt alle Formate ab

Texturhardware

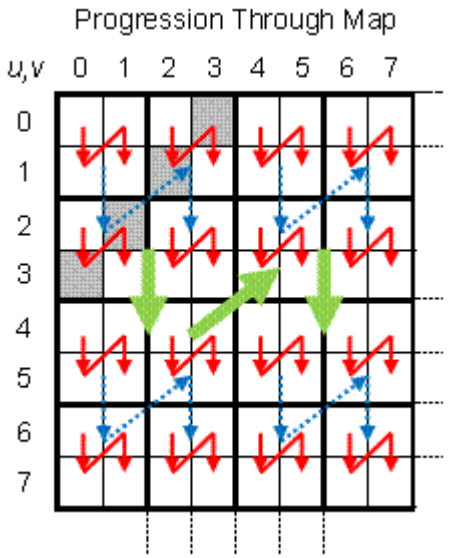
- Darstellung:
  - Linear
  - Swzizzled
  - Tiled

Linear

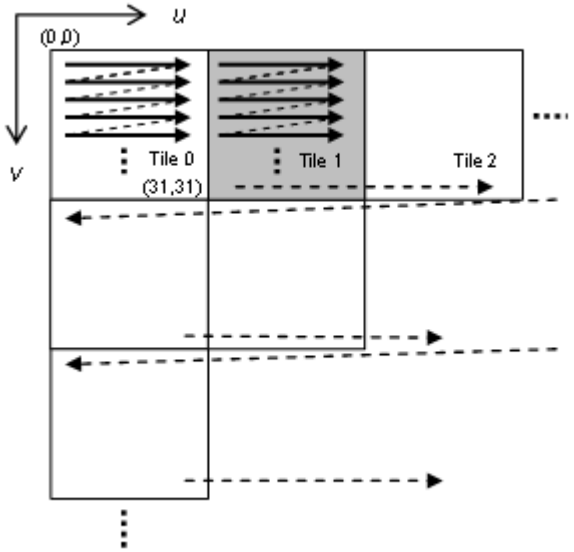


Texturhardware

- Swizzled

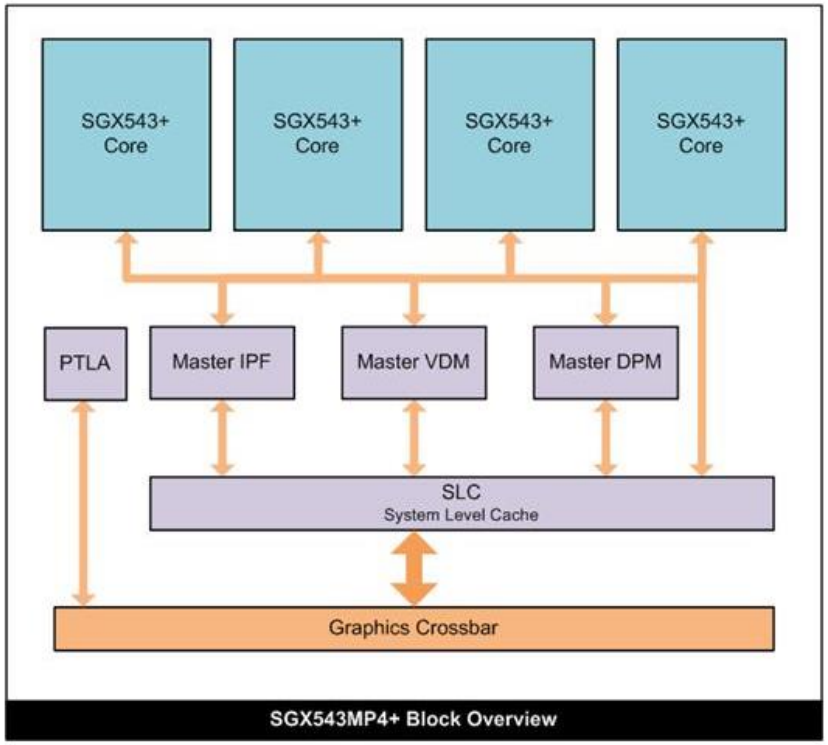


- Tiled



Texturhardware

- Vita hat PTLA (present and texture load accelerator)
- zum Transfer und Umcodierung der Texturen



LibGXT

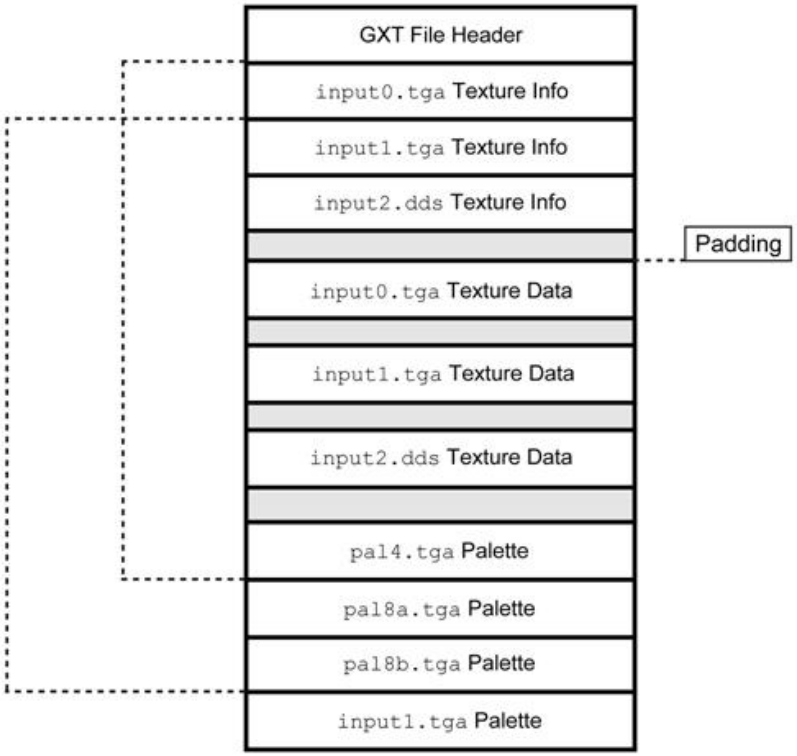
- Die Verwendung von Texturen wird durch LibGXT vereinfacht.

- 2 Komponenten:
  - Kommandozeilenwerkzeug:
    psp2gxt -i <input file(s)> -o output.gxt
  - Die eigentliche Library zum Verwalten der Daten

- Kommandozeilenwerkzeug:
  - Swizzled normalerweise die Daten
  - Verarbeitet .tga .dds und .pvr (Containerformat)

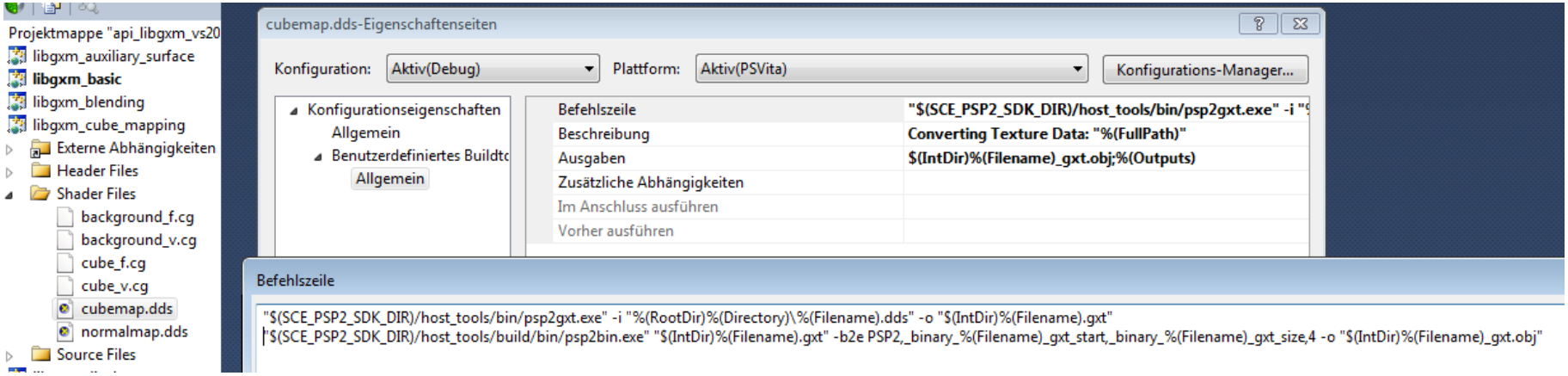
LibGXT

- Psp2gxt kann verschiedene Texturen zusammenfassen



LibGXT

- Texturen können entweder über den Compiler eingebaut werden oder klassisch geladen werden



LibGXT

- Verwendung der SceIoLib:
- In System/Kernel:

1. Öffnen der Datei.
2. Seek ans Ende ausführen (sceIoLSeek(df, 0, SCE_SEEK_END)) – liefert Dateigröße
3. Speicher anfordern
4. Datei laden

LibGXT

- Texturen müssen vor dem Zeichnen in Speicher kopiert werden:
  - Ungecachter Zugriff
  - Alignment: SCE_GXM_TEXTURE_ALIGNMENT
  - Für GPU mappen

- Gewinnung der Textur 0 aus Rohpointer gxt:

```
// Get the size of the texture data.
const uint32_t dataSize = sceGxtGetDataSize(gxt);

// Get pointer to start of the texture data.
const void *dataSrc = sceGxtGetDataAddress(gxt);

// Copy texture data to GPU mapped memory.
memcpy(textureData, dataSrc, dataSize);

// Set up the texture control words for texture 0.
SceGxmTexture texture;
sceGxtInitTexture(&texture, gxt, textureData, 0);
```

LibGXT

- SceGxmTexture ist opaque
- Init Texture Initialisiert eine Textur aus dem Block

```
#include <gxt.h>
inline SceGxtErrorCode sceGxtInitTexture(
    SceGxmTexture *texture,
    const void *gxt,
    const void *textureData,
    uint32_t textureIndex
);
```

LibGXT

- Texturfilter können danach gesetzt werden mit
  - sceGxmTextureSetMinFilter
  - sceGxmTextureSetMagFilter
- Beispiel beides mit
  - SCE_GXM_TEXTURE_FILTER_LINEAR

Texturen setzen

- Texturen werden vor dem Zeichnen gesetzt mit

```
SceGxmErrorCode sceGxmSetFragmentTexture(
    SceGxmContext *context,
    uint32_t textureIndex,
    const SceGxmTexture *texture
);
```

- Index ist die Texture stage die gebunden werden soll (0-15)

Benutzung im Fragmentshader

- Die Textur wird als uniform Variable definiert, wichtig: Bindungssematik
- uniform sampler2D normalTex: TEXUNIT0
- Textur auslesen mit: float4 tex2D(sampler2D samp, float2 s)
- Dies können Texturkoordinaten sein

Texturkoordinaten

- Texturkoordinaten werden an Fragmentshader mit Semantik TEXCOORD0, TEXCOORD1 übergeben

- Texturkoordinaten können
  - Im Fragementshader berechnet werden
  - Im Vertexshader berechnet werden
  - Mit der Vertexdefinition übergeben werden.

Zusammenfassung

1. Texturen mit **psp2gxt** erzeugen
2. Datei in den Speicher laden
3. Mit **sceGxtInitTexture** binden
4. Texturkoordinaten zur Verfügung stellen.
5. **uniform sampler2D** Variable in fragmentshader einbauen
6. Textur vor dem Zeichnen binden
   sceGxmSetFragmentTexture
7. Texturfarbe mit **tex2D** auslesen

Aufgabe

- **Aufgabe**: Nehmen Sie das erste Beispiel mit dem Dreieck her und versehen Sie das Dreieck mit einer beliebigen Textur.

Zusammenfassung

- Die Texturhardware der PSP
- Verwendung der LibGXT
  - Konvertieren von Texturen
  - Einbindung der Texturen

- Setzen der Texturen an den Fragmentshader
- Benutzung der Textur im Fragmentshader