Text rendering: Caching/rendering systems

Description

The rasterization of a glyph is a very costly operation. Paired with shaping and positioning algorithms, it becomes even more expensive. One strategy to counter this is a caching and rendering mechanism. A cache is a storage type which maps a key to a specific representation of a value. The transformation of the rasterization into the value of a cache entry might differ from cache type to cache type. Based on the difference an optimal render strategy has to be internally selected. This is the reason why the text renderer and the caching system are strongly related.

Candera supports different caching and rendering systems with different backgrounds and goals.

No Caching

The easiest form is to cache nothing. This strategy has very little performance but requires the least memory. Its strategy is to retrieve a rasterized glyph, blit it to the display and free all the memory content. This is a strategy which is only advised to use on very low end devices. Otherwise it is not recommended.

Bitmap Cache

Bitmap caching is a strategy to store all the glyphs of a single text within a bitmap. When the text or its properties changes, the bitmap has to be generated. This bitmap will be uploaded and stays uploaded. Every time the visual area of the text becomes invalidated but the text has not been changed, it will rerender the bitmap but not generate it newly. The bitmap is generated with a software blit and therefore generates CPU load.

Glyph Atlas

The glyph atlas is an uploaded texture which contains all cached glyphs. If a glyph is not available there, the texture will be updated by adding the rasterized glyph. A text change requires only an update of a vertex buffer which references the glyphs within the atlas. Therefore, glyphs do not have to be uploaded anymore as soon as they are in the cache. The glyph atlas has a learning curve to fill up the glyph atlas. This solution is a GPU load solution.

Glyph Cache

The glyph cache stores the rasterized glyphs. Which then will be uploaded and rendered in every render call. This is typically a performance bottleneck but useful for very specific use cases. It is not recommended to use glyph cache in common use cases.

Surface Cache

The surface cache uses the same glyph storage mechanism as glyph cache but renders it similar to the bitmap cache. However, surface cache does this on GPU side whilst bitmap cache does it on CPU side. A change within a text leads to a draw call for every single glyph. If the new text requires a larger area to display the text, a new FBO will be created. FBO creation is an expensive operation. This cache is made for specific hardware devics. It has more drawbacks than advantages in common use cases.

Cache Usages

This section describes where the cache types are available and what are recommended usages.

TextNode2D:

- Bitmap
- Glyph atlas
- Surface cache (Freetype only)
- Glyph/No cache (Freetype only)
- Glyph cache (Freetype only)

TextBrush:

- Bitmap
- Glyph atlas
- No cache (Freetype only)
- Glyph cache (Freetype only)
- Surface cache (Freetype only)

CanvasText:

- Glyph atlas

The recommended usages are bitmap cache as CPU solution and glyph atlas as GPU solution. The usage depends on the given project. All other cache types are written for specific use cases (e.g. specific devices).

Revision #4
Created 2023-02-22 01:46:58 UTC by Tsuyoshi.Kato
Updated 2023-06-13 05:24:35 UTC by Tsuyoshi.Kato