RENDERING ENGINE への道のり

今年やっていたこと

Rustでブラウザ風レンダリングエンジンを作ろうとしていた...

1. HTML/CSSパーサーを書いたり
2. スタイル計算アルゴリズムを実装したり
3. レンダリング処理の実装を目指したり

https://github.com/tetracalibers/learn-browsers-work

レンダリング処理

- どうせならGPUレンダリングを実装したい
- そろそろシェーダーでも遊びたい

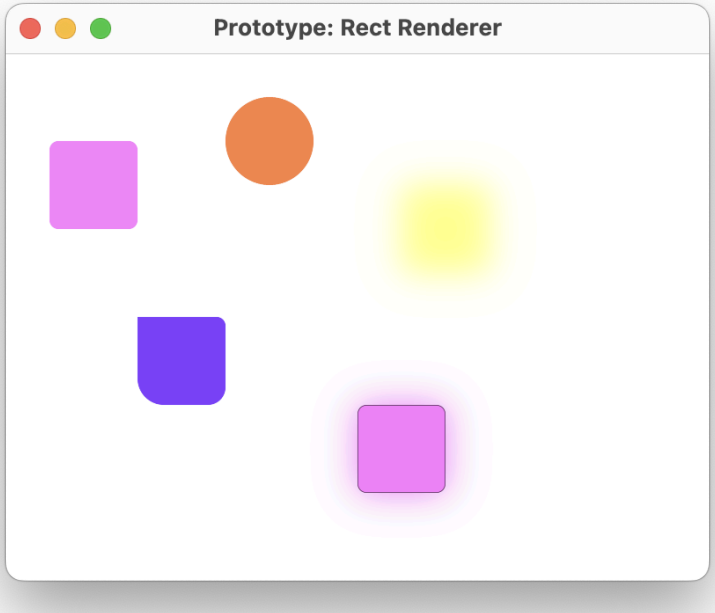
WebGPUの勉強を兼ねて、Rustのwgpuで

夏頃に作った試作品

- 長方形レンダリング
- テキストレンダリング

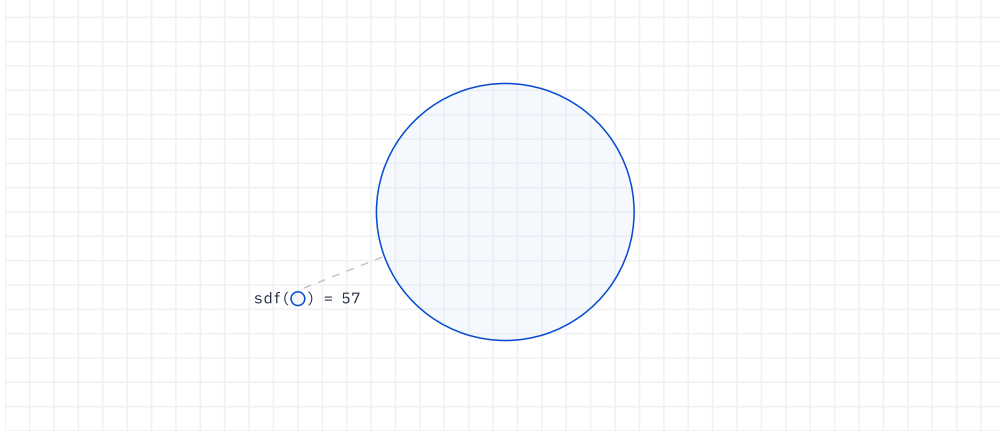
長方形レンダリング

https://github.com/tetracalibers/wgpu-practice-ground/pull/16



SDF（符号付き距離関数）

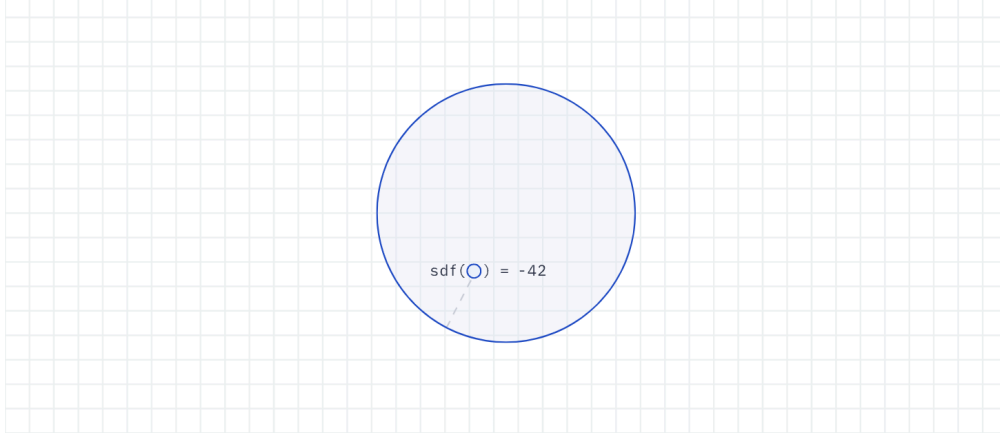
入力位置が与えられると、数学的に定義されたオブジェクトの端までの距離を返す関数



図形内の点のSDFは負の数

境界への距離が負の数

=> 内部にめり込んでいる状態



SDFを使ったGPUレンダリング

着想はZedを支えるUIライブラリGPUIから

https://zed.dev/blog/videogame

1. SDFを計算
2. SDFの符号で境界内かどうかを判定する
3. 境界内なら塗りつぶす...

という処理をシェーダーで行う

角丸長方形のドロップシャドウ

Figmaの共同創設者Evan Wallaceが開発した手法

https://madebyevan.com/shaders/fast-rounded-rectangle-shadows/

RED OTTER

https://www.red-otter.dev/

canvas要素内でWebのレンダリングを再現しようとするプロジェクト

テキストレンダリング

https://github.com/tetracalibers/wgpu-practice-ground/pull/18

SDFを使ったテキストレンダリング

1. フォントファイルから各文字を取り出す
2. 取り出した文字をテクスチャに格納（アトラス）
3. アトラス内の各グリフをSDF化
4. アトラス画像をシェーダーに送って描画

SDFフォントはスケーラブル

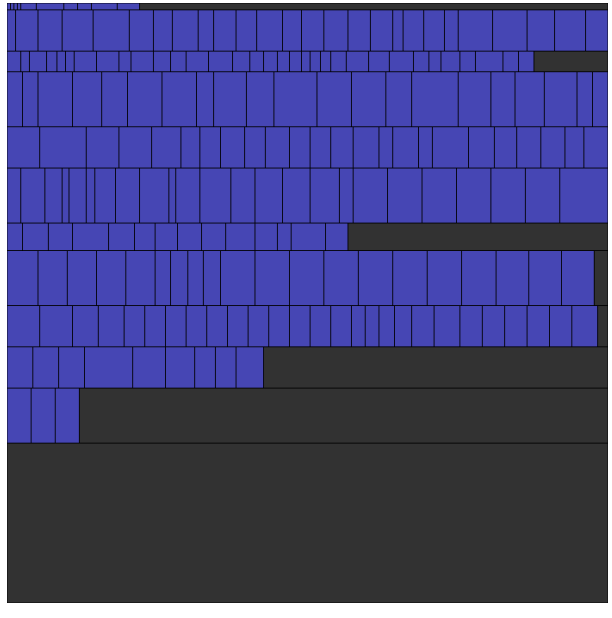
拡大しても鮮明（右から2番目）

https://libgdx.com/wiki/graphics/2d/fonts/distance-field-fonts

アトラスの生成

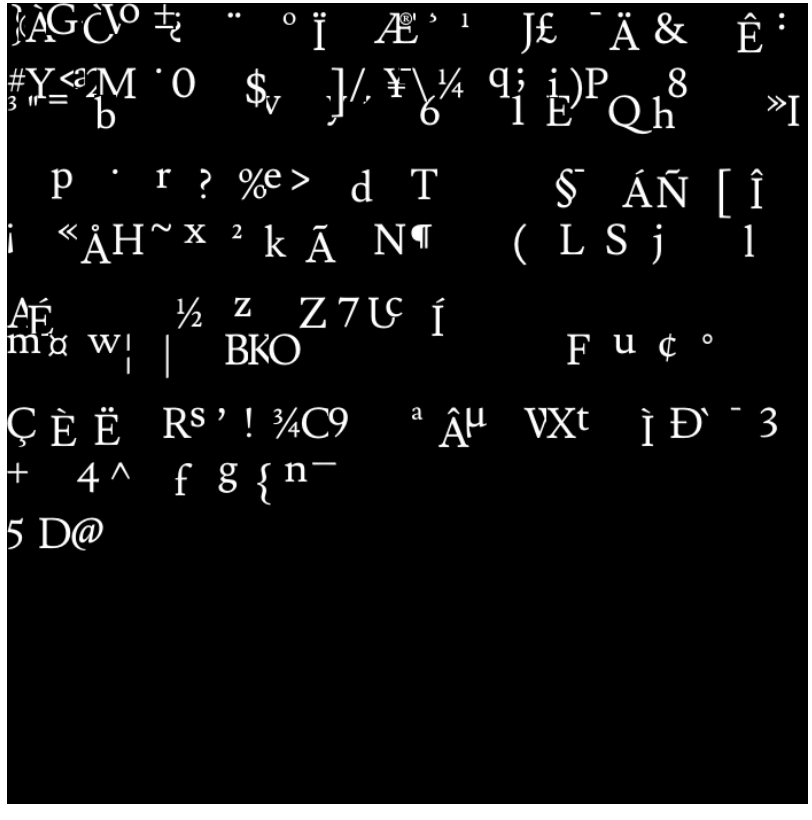
ビンパッキング アルゴリズム

効率的な矩形の格納を行うアルゴリズム



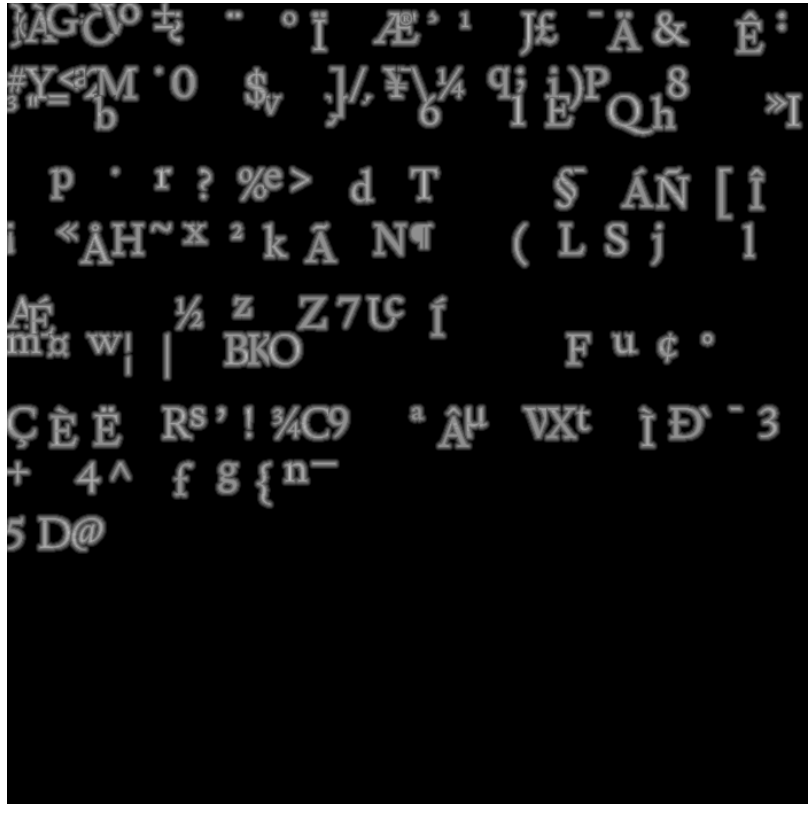
etagereで実装

矩形にグリフを配置



さらにSDF化

各ピクセルはグリフの端からの距離を表す



アトラスを使ったレンダリング

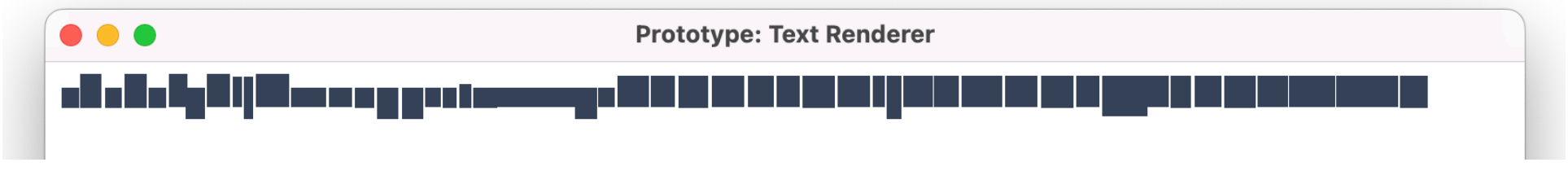
アトラス≒スプライト画像

シェーダーでは、アトラスから描画したい文字の
SDFを取り出してレンダリング

```
@fragment
fn fs_main(in: VertexOutput) -> @location(0) vec4f {
  let g = text.glyphs[in.instance];

  // textureSample.a : 矩形を描画
  // textureSample.r : 文字を描画
  let distance = textureSample(atlas, atlas_sampler, in.uv).r;

  var width = mix(0.4, 0.1, clamp(g.font_size, 0.0, 40.0) / 40
  width /= 2.0; // TODO: apply dpr
  let alpha = g.color.a * smoothstep(0.5 - width, 0.5 + width,

  return vec4f(g.color.rgb, alpha);
}
```

ちなみにSDFのALPHA値を取り出すと...

グリフの占めるスペースが矩形で現れる



これからは？

これからも...

細々とレンダリングエンジンを作っていきたい

- Flutterのアーキテクチャも参考にしつつ
- シェーダーでどこまでできるか？の実験は続く
- （マルチスレッドも勉強しないと...）
- いつかGrid Layoutの実装とかもやってみたい

目指すは自作ブラウザ...なのかはまだわからない

FIN.