Akademy

Closing the Distance Between CPU and GPU with Signed Distance Fields

Arjen Hiemstra

Next up...

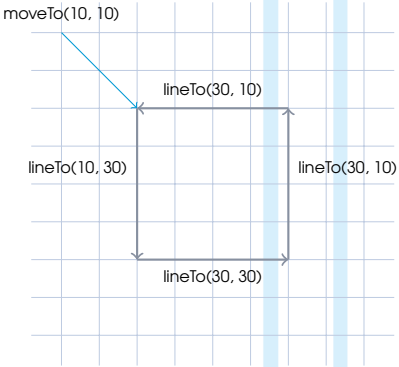
1 Some Concepts

2 Signed Distance Fields

3 Use Cases

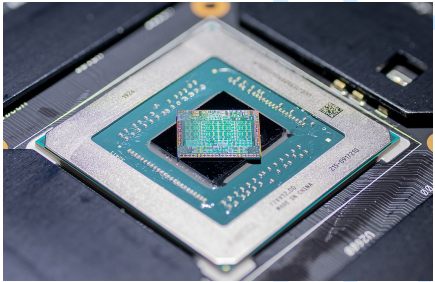
2D Rendering: The Postscript Model

- Dominant 2D rendering model for at least 30 years.
- Essentially models a 2D plotter.
- Used by major APIs like QPainter and Canvas.



3D Rendering: The GPU

- Pratically every modern computer has one.
- Massively parallel.
- Originally designed to accelerate 3D.



Programmable Shaders

- Small programs that get executed by the GPU.
- Written in GLSL or other shading language.
- Various stages exist corresponding to different parts of pipeline.

```
// Vertex Shader
uniform mat4 modelViewProjection;
in vec4 vertexPosition;
void main()
{
    gl_Position = vertexPosition *
        modelViewProjection;
}

// Fragment Shader
out vec4 color;
void main()
{
    color = vec4(1.0, 0.0, 0.0, 1.0);
}
```

Next up...

1 Some Concepts

2 Signed Distance Fields

3 Use Cases

The Problem

It would be nice if we could use the GPU to accelerate rendering complex 2D items.

But The PostScript model is inherently serial, while the GPU is massively parallel.

So We need a new rendering model for 2D shapes.

Example: Circle

$$sdf(\vec{p}) = \|\vec{p}\| - r$$

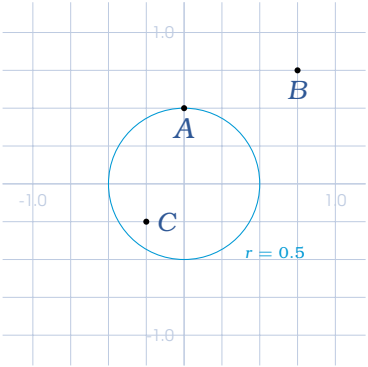
Where:

$\vec{p}$ is a 2D vector in range $\langle -1, 1 \rangle$

r is the radius of the circle.

Turning Math to Image

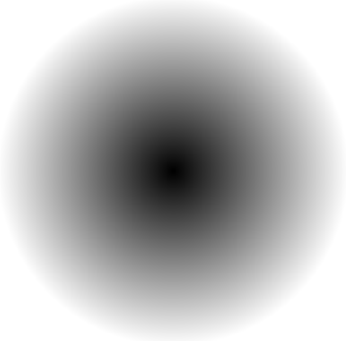
$$sdf(\vec{p}) = \|\vec{p}\| - r$$



A	$\vec{p} = \begin{matrix} 0.0 \\ 0.5 \end{matrix}$	Distance = 0.0
B	$\vec{p} = \begin{matrix} 0.75 \\ 0.75 \end{matrix}$	Distance = 0.56
C	$\vec{p} = \begin{matrix} -0.25 \\ -0.25 \end{matrix}$	Distance = -0.15

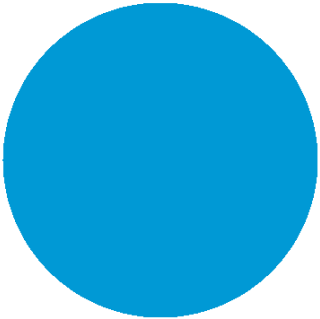
On the GPU

```
uniform float radius;
uniform vec4 color;

in vec2 uv;

out vec4 out_color;

void main()
{
    float distance = length(uv) - radius;
    out_color = vec4(distance);
}
```



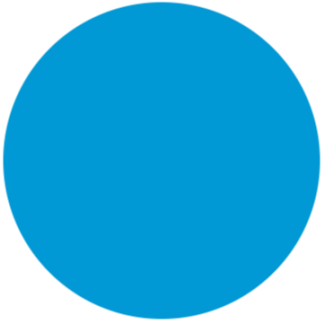
An Actual Circle

```
uniform float radius;
uniform vec4 color;

in vec2 uv;

out vec4 out_color;

void main()
{
    float distance = length(uv) -
        radius;
    out_color = distance > 0.0 ?
        vec4(0.0) : color;
}
```

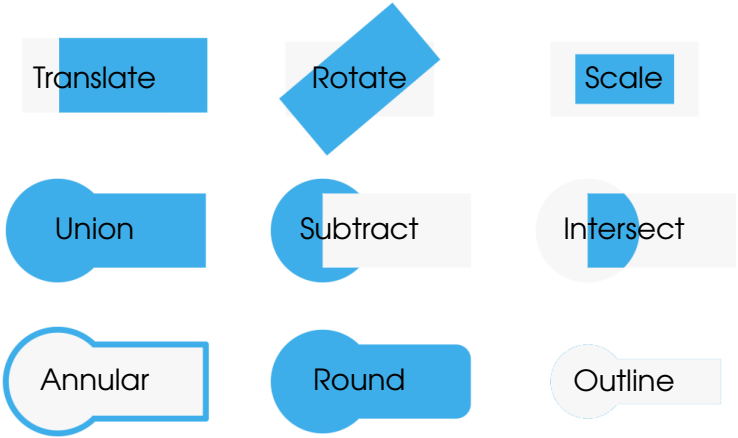


Antialiasing

```
uniform float radius;
uniform vec4 color;

in vec2 uv;

out vec4 out_color;

void main()
{
    float distance = length(uv) - radius;
    out_color = mix(color, vec4(0.0), clamp(distance * 100.0, 0.0, 1.0));
}
```



Operations


Translate
Rotate
Scale
Union
Subtract
Intersect
Annular
Round
Outline

Next up...

1 Some Concepts

2 Signed Distance Fields

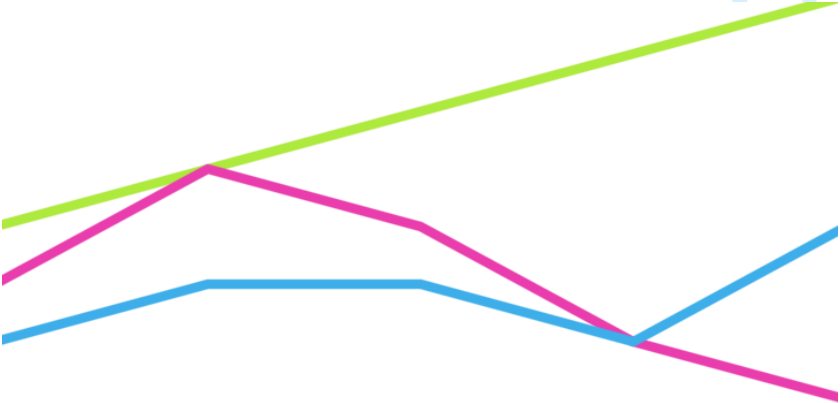
3 Use Cases

Rendering Circles

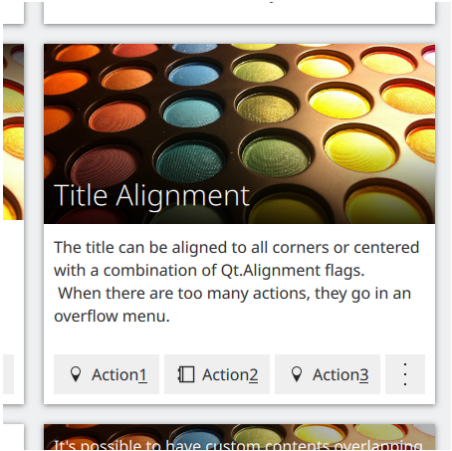
- Rendering circles is a tricky problem in 3D graphics.
- Standard geometric approach provides an approximation.

Pie Charts

Line Charts



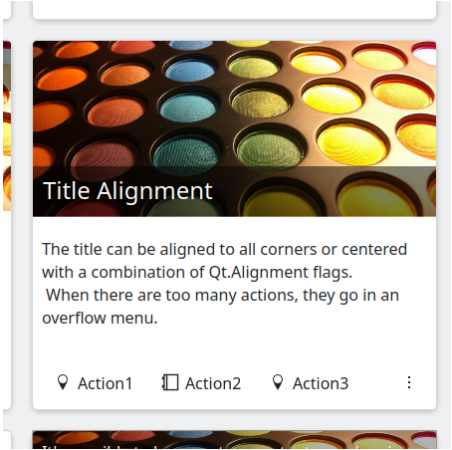
Kirigami's Cards



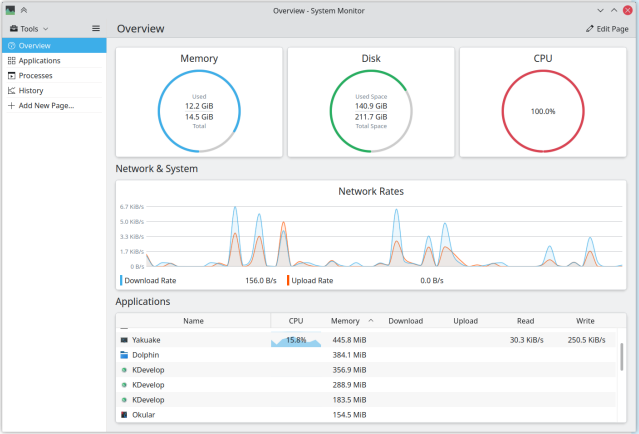
ShadowedRectangle

ShadowedTexture

New Cards


Title Alignment
The title can be aligned to all corners or centered with a combination of Qt.Alignment flags.
When there are too many actions, they go in an overflow menu.
Action1
Action2
Action3

Putting It Together



Closing

Special Thanks To:
Inigo Quilez
https://iquilezles.org/

Links

KQuickCharts	https://invent.kde.org/framework/kquickcharts/
Kirigami	https://invent.kde.org/framework/kirigami/
Author	Arjen Hiemstra Matrix: @ahiemstra:kde.org