The Unusual Rendering Pipeline of Sigils – Battle for Raios

Dietmar Hauser
Head of Console Technology
Sproing Interactive Media GmbH

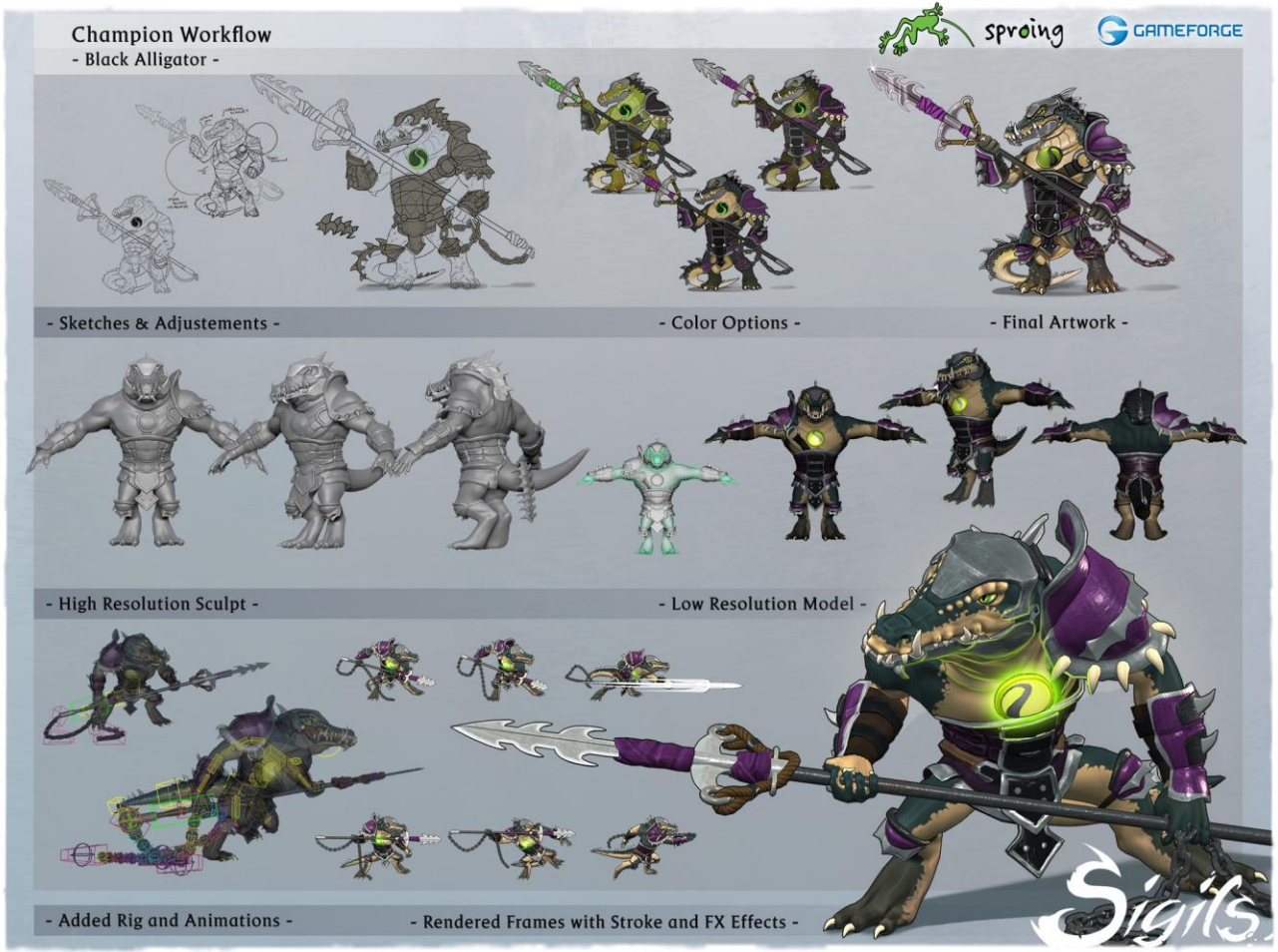
GAME DEVELOPERS CONFERENCE™ EUROPE
CONGRESS-CENTRUM OST KOELNMESSE · COLOGNE, GERMANY
AUGUST 11–13, 2014 · EXPO: AUGUST 11–12, 2014

About „Sigils – Battle For Raios“

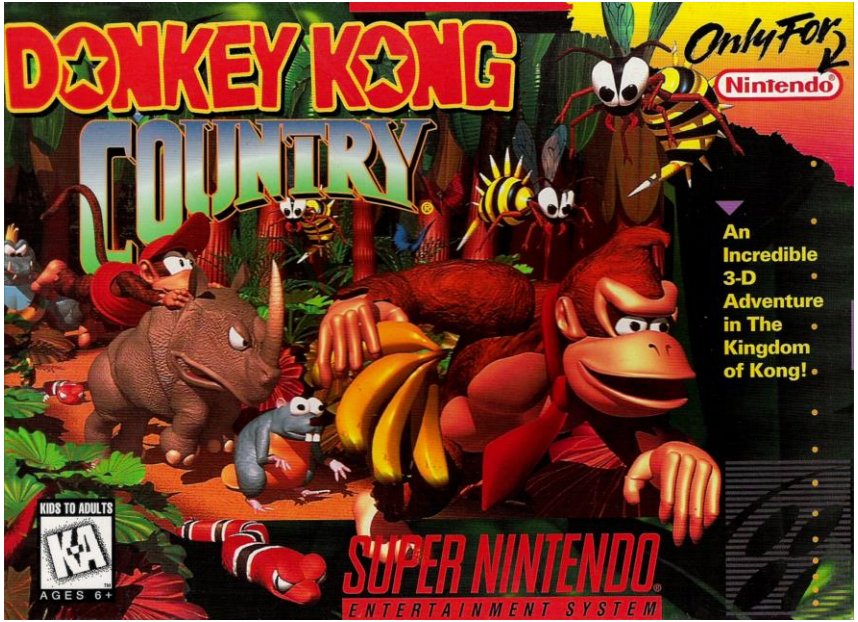
- Tablet only
- MOBA / RPG
- Realtime Multiplayer
- PvE and PvP
- Free To Play


ramyou
ninor
1
4
4
vangelic
ramyou
ninor
2
7

The Evolution of the Visual Style


Champion Workflow
- Black Alligator -
sproing
GAMEFORGE
- Sketches & Adjustements -
- Color Options -
- Final Artwork -
- High Resolution Sculpt -
- Low Resolution Model -
- Added Rig and Animations -
- Rendered Frames with Stroke and FX Effects -
Sigils

A little history...



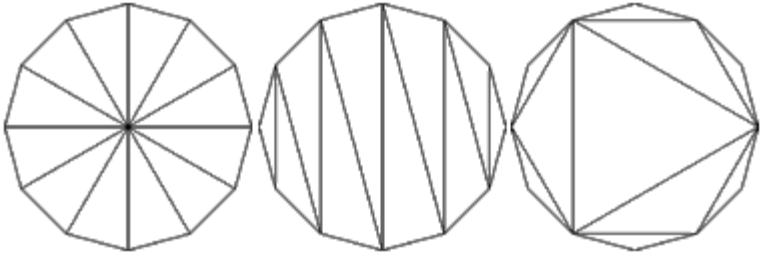
About Mobile GPUs

Mobile GPU Challenges

- Low fill rate
- High pixel count
- Tile-based deferred rendering
  - Complicates resource management
  - Render target switches are expensive
- OpenGL ES
  - Familiar, but not well suited

Pre-rendered Sprites

- Reduce Overdraw
  - Quads are very wasteful
  - Find outline
    - Using a convex hull algorithm
  - Tessellate smartly
    - "Greedy" algorithm
    - http://www.humus.name



Pre-rendered Sprites

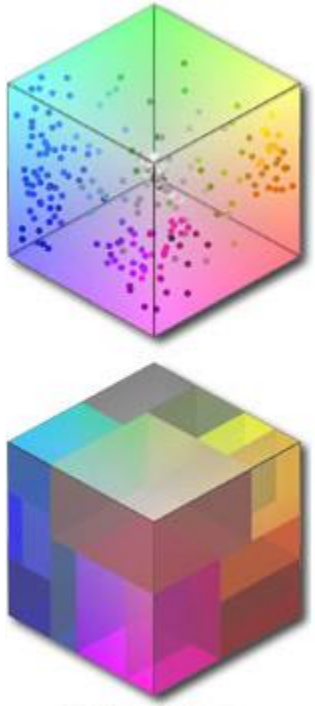
- Reducing Memory
  - Only one frame is needed at a time
  - “Memory to memory streaming”
  - Frequently used frames are cached
  - Compression algorithm is important
    - DEFLATE vs. LZ4

Pre-rendered Sprites

- GPU Texture Compression
  - Reliably reduces texture size
  - Net performance gain (!)
  - Visible artefacts on 2D images
  - Fragmented on mobile devices
    - PVRTC vs. ETC vs. S3TC

Pre-rendered Sprites

- Color Quantization
  - Split image
    - Index texture: I8
    - Palette texture: RBGA8
  - Reassemble in Pixel Shader
    - Sample with point filtering
    - Use as index to sample in palette
    - 4 times for linear filtering



Pre-rendered Sprites

```
// read indices
float4 index = float4(  tex2D(sourceTex, In.TexCoord0.xy).r,
                        tex2D(sourceTex, In.TexCoord0.zw).r,
                        tex2D(sourceTex, In.TexCoord1.xy).r,
                        tex2D(sourceTex, In.TexCoord1.zw).r);
// nudge indices to center of texel
index = index * (paletteWidth-1.0f)/paletteWidth + 0.5f/paletteWidth;
// sample colors
float4 textureColor0 = tex2D(paletteTex, float2(index.x, paletteIndex));
float4 textureColor1 = tex2D(paletteTex, float2(index.y, paletteIndex));
float4 textureColor2 = tex2D(paletteTex, float2(index.z, paletteIndex));
float4 textureColor3 = tex2D(paletteTex, float2(index.w, paletteIndex));
// bilinear filtering
float2 lerps = frac(In.TexelPos);
float4 textureColor01 = lerp(textureColor0, textureColor1, lerps.x);
float4 textureColor23 = lerp(textureColor2, textureColor3, lerps.x);
float4 textureColor = lerp(textureColor01, textureColor23, lerps.y);
// and we're done
return textureColor;
```

Pre-rendered Sprites

Original Image

Quantized

Pre-rendered Sprites

- Dealing with Alpha
  - Reduces palette colors
  - Causes sporadic see through spots
- Separate it!
  - Index texture: IA8
  - Palette texture: RGBA8

Pre-rendered Sprites

Original Image

Quantized + Alpha

Pre-rendered Sprites

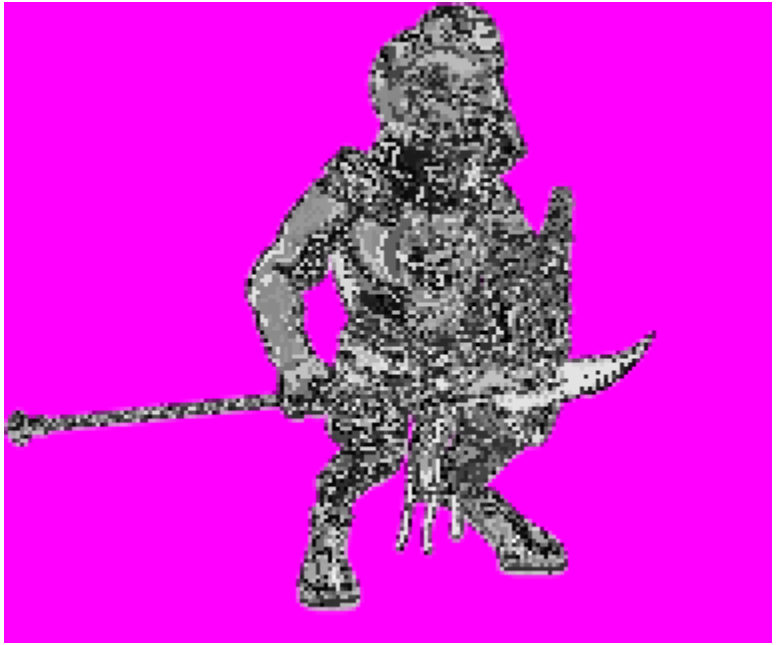
- Error Diffusion
  - Aka dithering
  - Many algorithms to choose from
  - Floyd-Steinberg seems most smooth

Pre-rendered Sprites

Original Image

Final Image

Pre-rendered Sprites



Post Processing

- The traditional approach
  - Bloom to make things glow
  - Color Grading to make things moody
  - Vignette to soften the corners
  - Works everywhere, right?
  - Not quite…
    - ~30ms for post fx alone

Post Processing

- Why so bad?
  - Low fill rate, high pixel count
  - Post FX are inherently full screen
  - Render target switching extremely costly
  - Zero benefits from tiled rendering

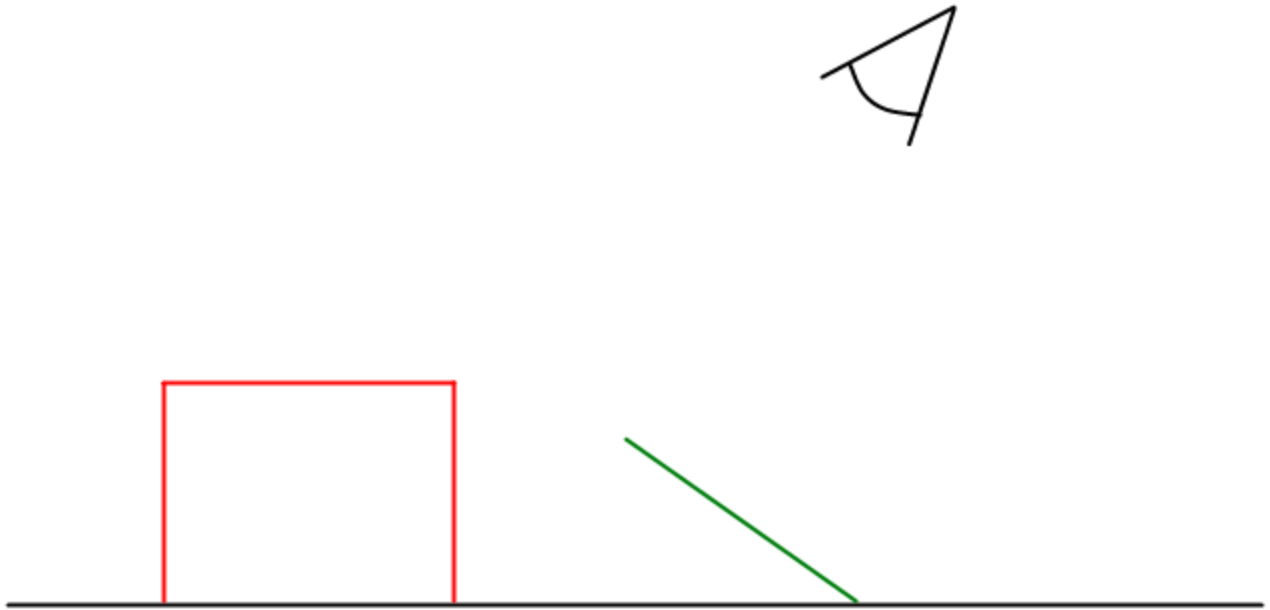
Post Processing

- A simplified approach
  - Replace bloom with flares or particles
  - One pass for color correction
    - Scene is rendered to render target
    - Then composited into back buffer
  - Vignette on screen borders only
    - ~80% faster than full screen

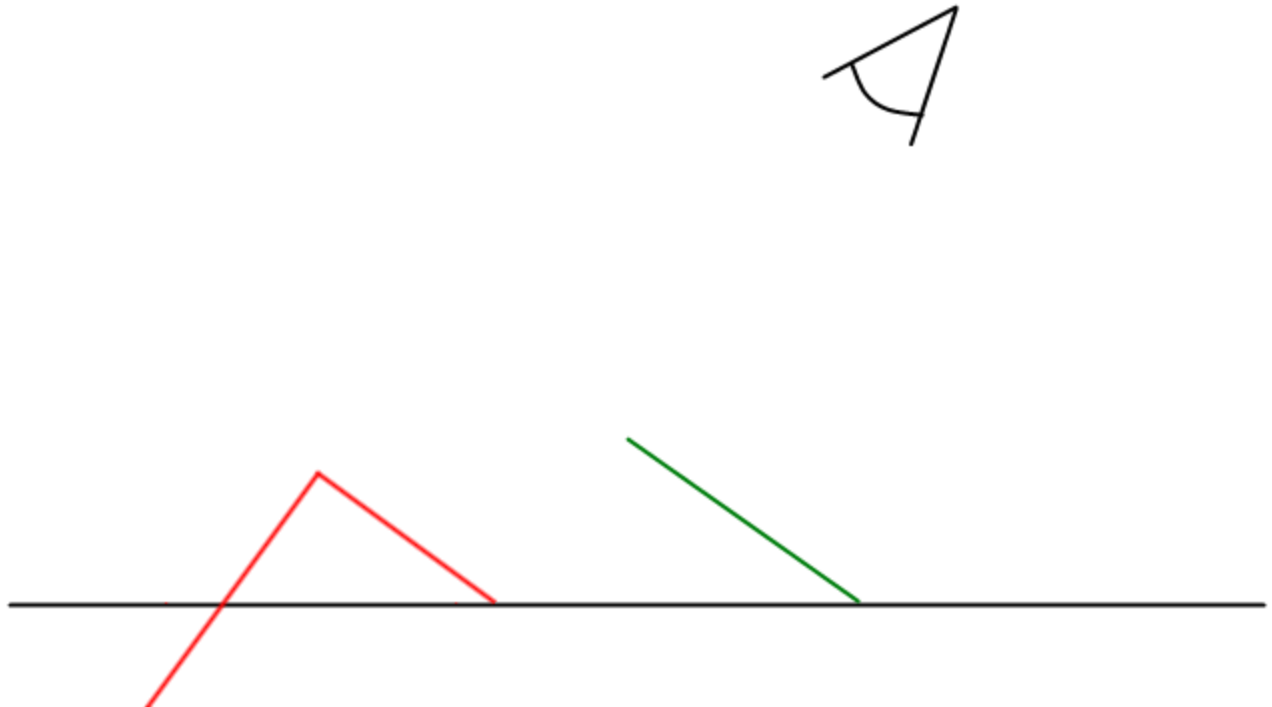
Scene composition

- Combining 2D sprites with 3D objects
  - 2D sprites are sorted back to front
  - Doesn’t work for 3D objects with depth
  - Use depth buffer instead
    - Render 3D object, write depth
    - Render 2D sprites, test depth
  - This leaves one issue…

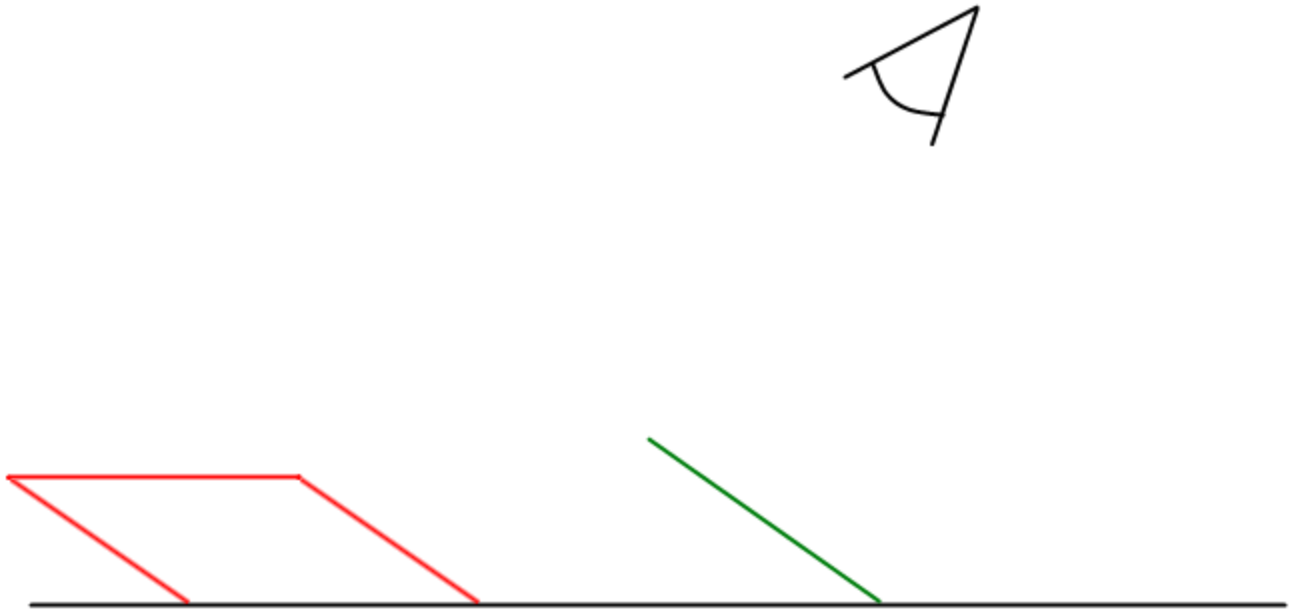
Scene composition



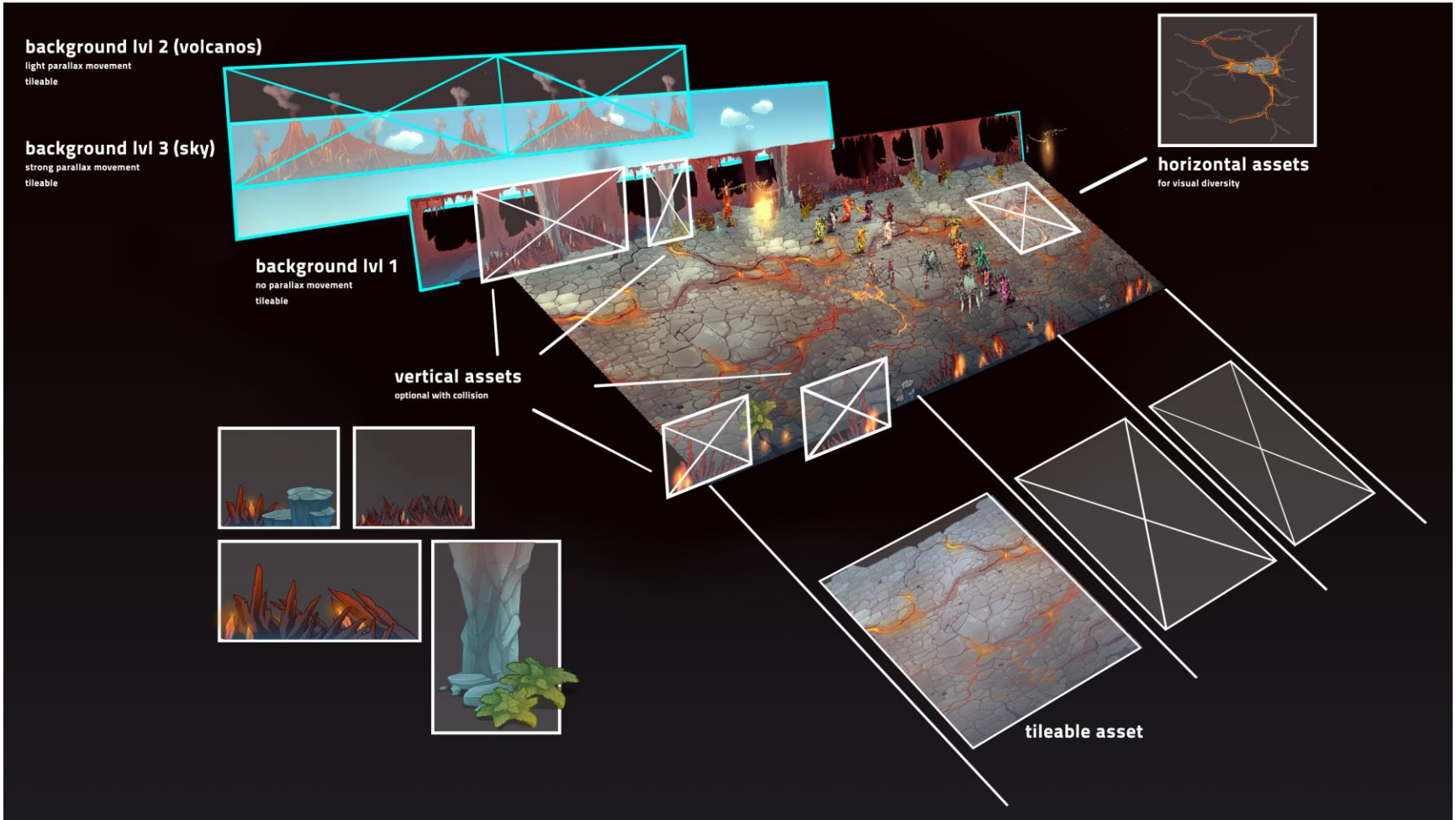
Scene composition



Scene composition



Scene composition


background lvl 2 (volcanos)
light parallax movement
tileable
background lvl 3 (sky)
strong parallax movement
tileable
background lvl 1
no parallax movement
tileable
horizontal assets
for visual diversity
vertical assets
optional with collision
tileable asset


ohmygoat
fridolin81
chickyka1
fridolin81
ohmygoat
2
17

Coming Soon!

http://en.sigils.gameforge.com

dietmar.hauser@sproing.com
@Rattenhirn

http://www.sproing.com
http://fb.me/sproing
@SproingGames

http://www.gameforge.com
http://fb.me/gameforge
@Gameforge

GAMEFORGE

sproing