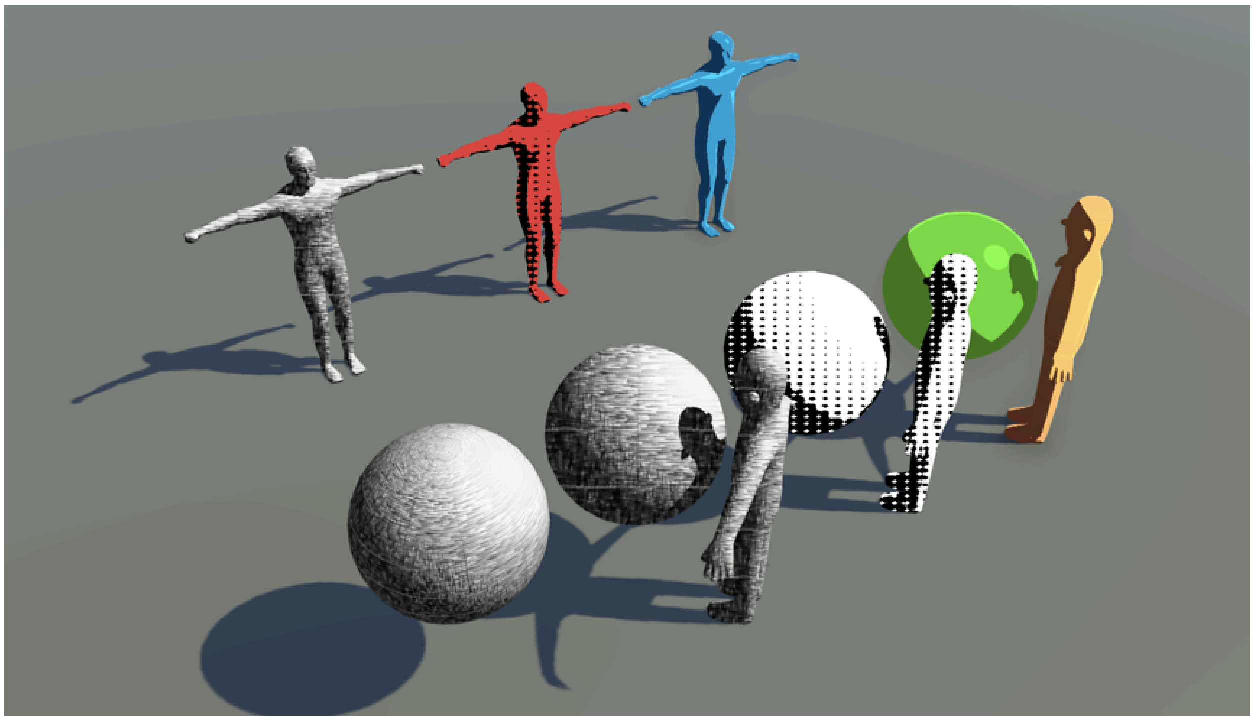
Ana Nguyen

SHADE-y business



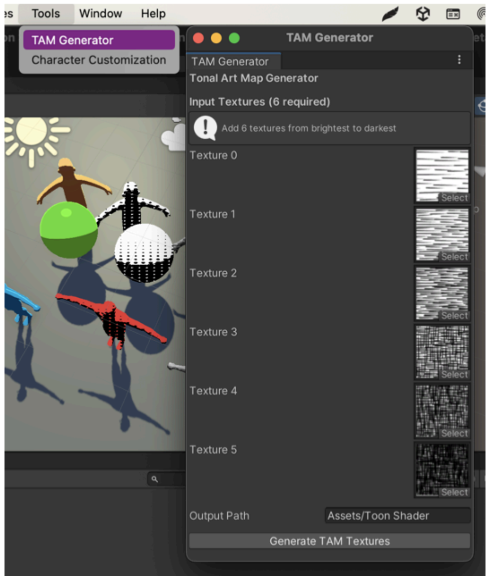
Code: https://github.com/anaxrocks/Cartoon-Shaders

TAM Generation

I referenced this paper https://hhoppe.com/hatching.pdf to implement tonal art maps for real-time cross hatching. An issue I came across with a cross-hatching shader is that it required a lot of texture lookups (my original had 6), so I was able to consolidate it into 2 texture maps.

This takes in six individual hatching textures (different tonal densities from bright to dark) and combines them into 2 consolidated texture maps. Each original texture represents a single hatching density and each RGB channel in the output textures stores a different hatching level, so Hatch0 stores levels 0-2 (lighter tones) and Hatch1 stores levels 3-5 (darker tones). This allowed me to optimize texture memory usage and simplify the CrossHatch shader access during rendering.

- Creates an editor window and functions as a menu item under the “Tools” tab for easy access.

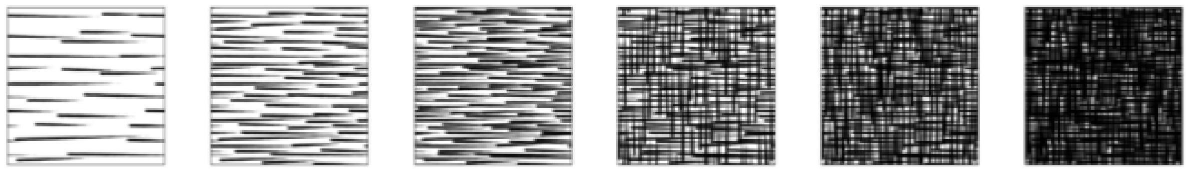


Add 6 textures from brightest to darkest

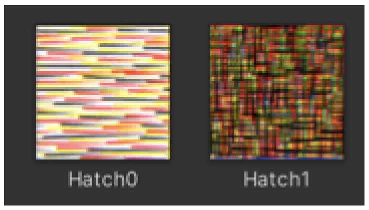
Output Path Assets/Toon Shader

- Texture Processing Pipeline
  - Input validation for presence and dimensional consistency
  - Per-pixel color channel packing from grayscale inputs to RGB outputs
  - PNG compression and file system integration
  - Asset database refresh to ensure immediate usability
- Import Settings
  - Bilinear filtering for smoother transitions
  - Mipmapping for LOD support
  - Repeat wrapping for seamless tiling
  - sRGB color space for proper rendering

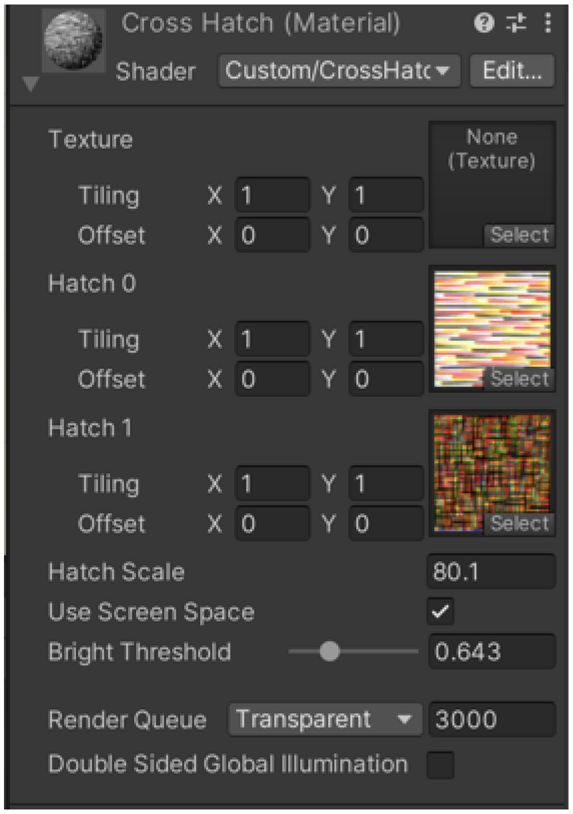
Input (6 original cross hatching textures referenced from the paper above)

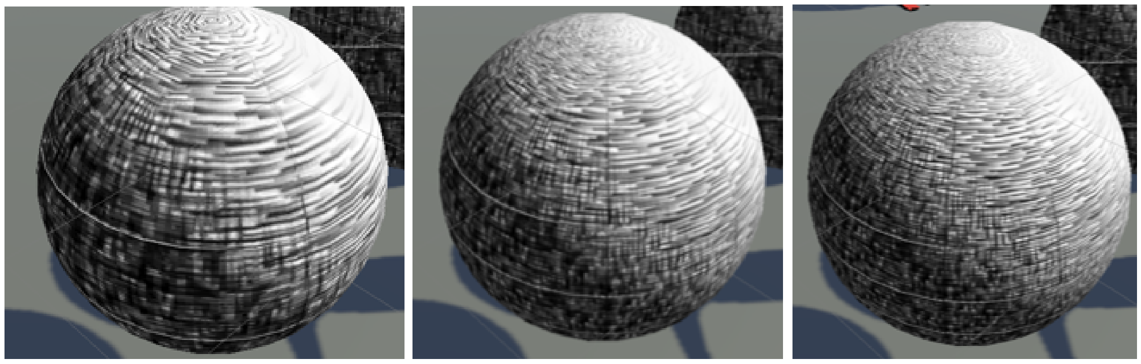


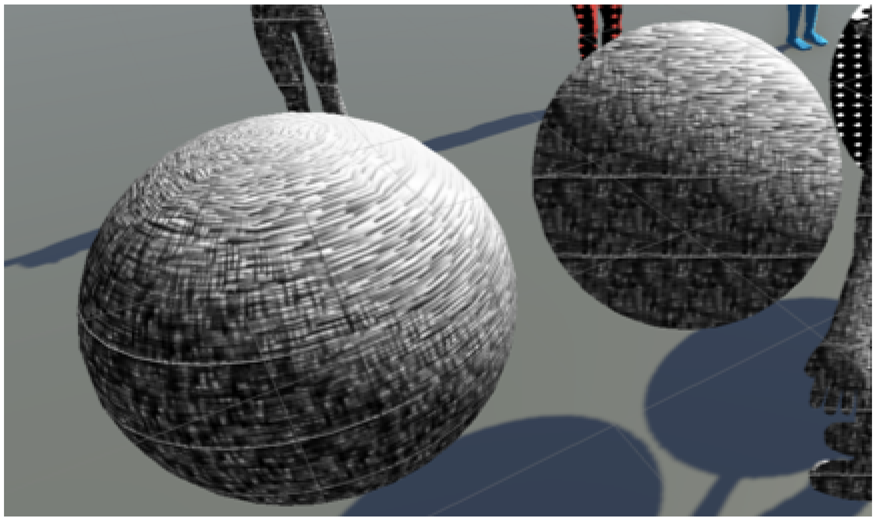
Output (what I ended up using for the crosshatch shader)



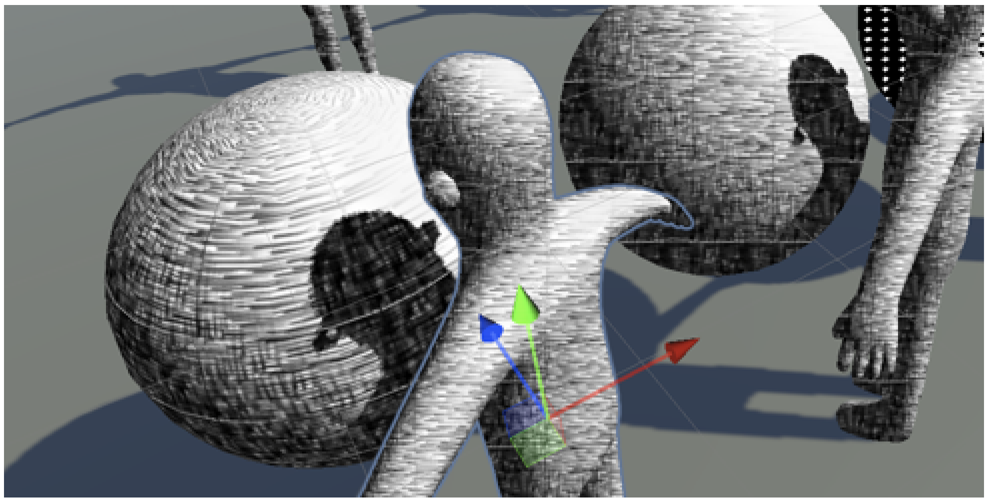
CrossHatch shader


Cross Hatch (Material)
Shader Custom/CrossHatc Edit...
Texture None (Texture)
Tiling X 1 Y 1
Offset X 0 Y 0
Hatch 0
Tiling X 1 Y 1
Offset X 0 Y 0
Hatch 1
Tiling X 1 Y 1
Offset X 0 Y 0
Hatch Scale 80.1
Use Screen Space ✓
Bright Threshold 0.643
Render Queue Transparent 3000
Double Sided Global Illumination

- Hatching () function takes a single intensity value and turns it into 6 different weights. The weight calculation creates an effect of progressively adding more hatching lines as the surface gets darker. The overbright calculation ensures that the brightest areas appear as pure white (rather than a variation of hatching). I referenced Kyle Halladay's weighting function for cross hatching for this shader. I also added a Hatch Scale to change the size of the cross hatching

Scale of 5, 8, 10
- UV Space handling: There's a screen space toggle.
In screen space mode, hatching appears fixed to the screen, creating a consistent hatching size regardless of object distance (like traditional illustration)
In UV space mode, hatching is attached to the object's surface, moving and scaling with the object

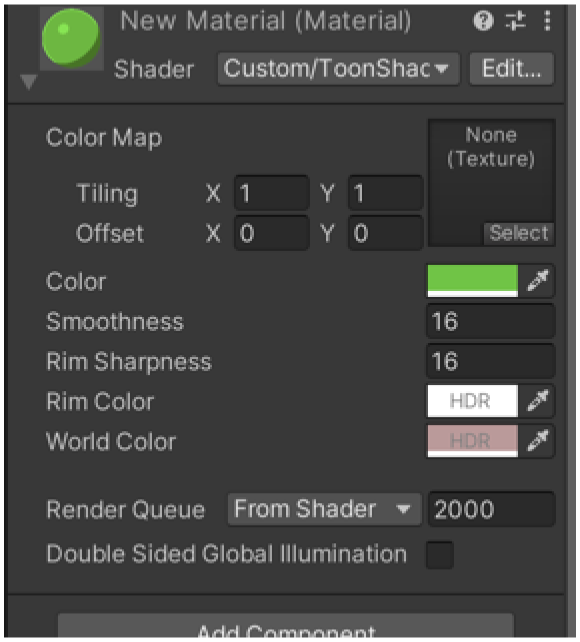

(left UV, right screen space)

- Proper shadow casting/receiving
  The cross-hatching effect properly responds to Unity's lighting environment, including shadows cast by other objects and shadow cascade systems for larger environments.



- Lighting calculations frag() function.
  Use the base texture color as part of intensity calculation to allow colored objects to maintain tonal relationships. Diffuse lighting/light color/shadow attenuation in calculations. RGB to luminance conversion.

Toon cel shading

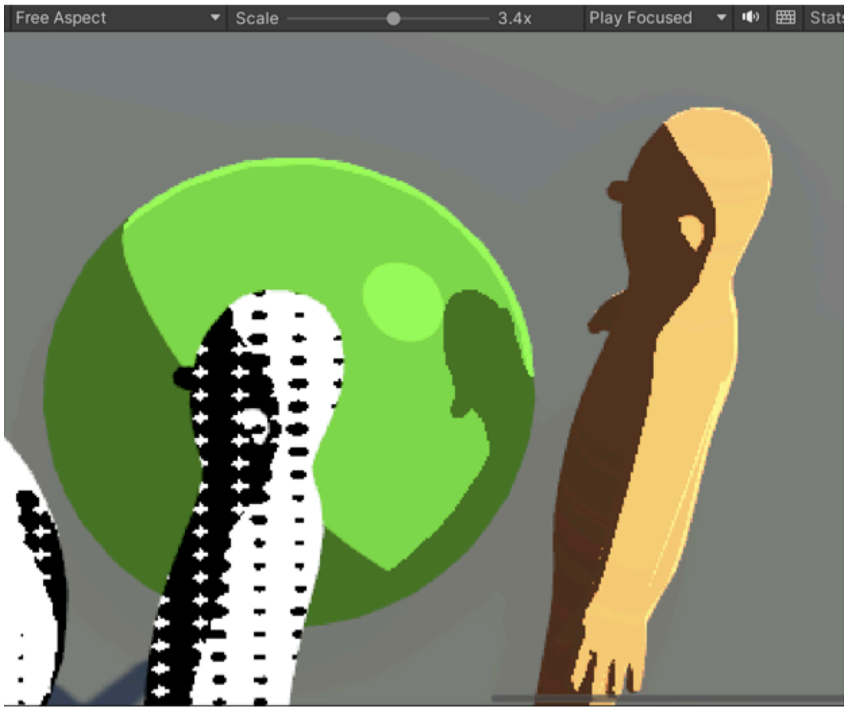

New Material (Material)

Shader: Custom/ToonShac | Edit...

Color Map	None (Texture)
Tiling	X 1 Y 1
Offset	X 0 Y 0
Color	🟩
Smoothness	16
Rim Sharpness	16
Rim Color	HDR
World Color	HDR
Render Queue	From Shader 2000
Double Sided Global Illumination	☐

Add Component

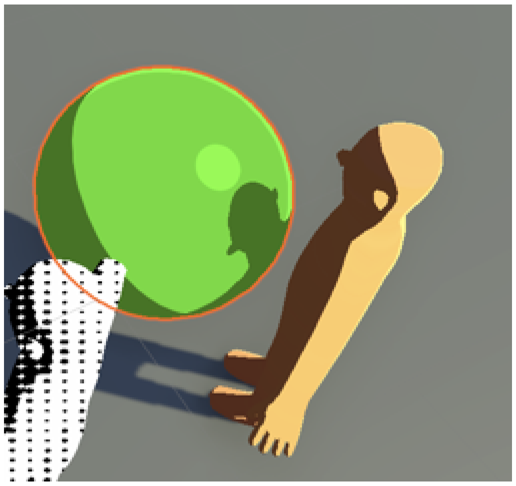
- easysmoothstep() function makes hard transitions between light and shadow areas.



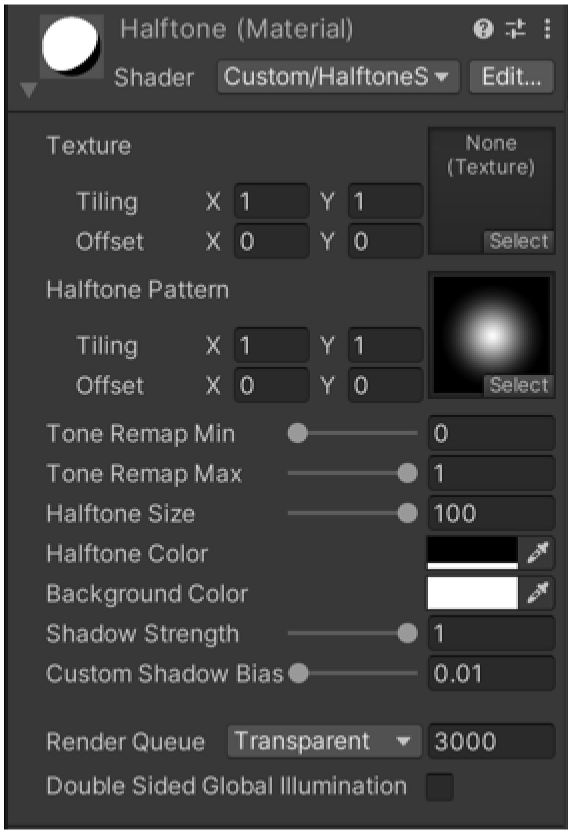
Zoomed in at scale 3.4. Way smoother than the original pixelated edges at normal scale.

- Rim lighting: Adds edge highlighting which is controlled by _RimSharpness and _RimColor parameters.
- Specular Highlights: Specular reflection with the _Smoothness parameter.

- Shadow Casting/Receiving: URP shadow cascades and soft shadows



Halftone Shading



Halftone (Material)

Shader Custom/HalftoneS Edit...

Texture: None (Texture)

Tiling X 1 Y 1

Offset X 0 Y 0

Halftone Pattern

Tiling X 1 Y 1

Offset X 0 Y 0

Tone Remap Min 0

Tone Remap Max 1

Halftone Size 100

Halftone Color

Background Color

Shadow Strength 1

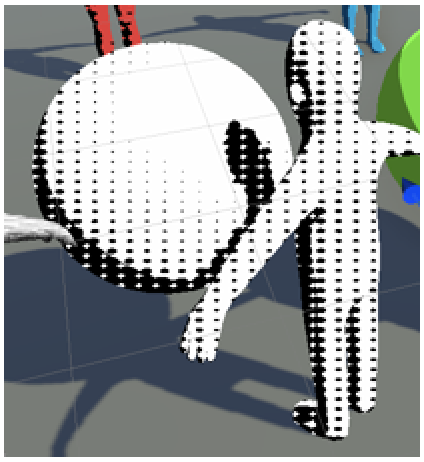
Custom Shadow Bias 0.01

Render Queue Transparent 3000

Double Sided Global Illumination

- Pattern-based rendering: Uses a provided texture to create the halftone effect
- Tone remapping: Controls the intensity range with _RemapMin and _RemapMax
- Screen-space pattern: Applies the pattern in screen space for consistent size
- Color: You can separate colors for dots and the background and specify which one you would like

- Shadow Casting/Receiving: URP shadow cascades and soft shadows. Change shadow intensity with _ShadowStrength



- Custom Shadow bias: Control over shadow rendering with _CustomShadowBias