

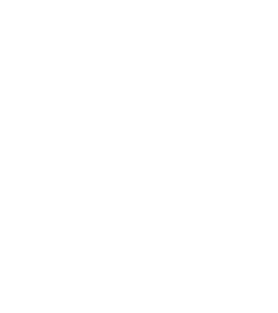
depth

An object that stores the framebuffer's depth data.

Each framebuffer uses a `WebGLTexture` object internally to store its depth data. The `myBuffer.depth` property makes it possible to pass this data directly to other functions. For example, calling `texture(myBuffer.depth)` or `myShader.setUniform('depthTexture', myBuffer.depth)` may be helpful for advanced use cases.

Note: By default, a framebuffer's y-coordinates are flipped compared to images and videos. It's easy to flip a framebuffer's y-coordinates as needed when applying it as a texture. For example, calling `plane(myBuffer.width, -myBuffer.height)` will flip the framebuffer.

Examples



// Note: A "uniform" is a global variable within a shader program.

```
// Create a string with the vertex shader program.
// The vertex shader is called for each vertex.
let vertSrc = `
precision highp float;
attribute vec3 aPosition;
attribute vec2 aTexCoord;
uniform mat4 uModelViewMatrix;
uniform mat4 uProjectionMatrix;
varying vec2 vTexCoord;

void main() {
    vec4 viewModelPosition = uModelViewMatrix * vec4(aPosition,
1.0);
    gl_Position = uProjectionMatrix * viewModelPosition;
    vTexCoord = aTexCoord;
}
`;

// Create a string with the fragment shader program.
// The fragment shader is called for each pixel.
let fragSrc = `
precision highp float;
```

Related References

[autoSized](#)
Toggles the framebuffer's autosizing mode or returns the current mode.

[begin](#)
Begins drawing shapes to the framebuffer.

[color](#)
An object that stores the framebuffer's color data.

[createCamera](#)
Creates a new p5.Camera object to use with the framebuffer.

