This help only covers the parts of GLSL ES that are relevant for Shadertoy. For the complete specification please have a look at GLSL ES specification

Language:

- **Version:** WebGL 2.0
- **Arithmetic:** ( ) + - ! * / %
- **Logical/Relatonal:** ~ < > <= >= == != && ||
- **Bit Operators:** & ^ | << >>
- **Comments:** // /* */
- **Types:** void bool int uint float vec2 vec3 vec4 bvec2 bvec3 bvec4 ivec2 ivec3 ivec4 uvec2 uvec3 uvec4 mat2 mat3 mat4 mat?x? sampler2D, sampler3D, samplerCube
- **Format:** float a = 1.0; int b = 1; uint i = 1U; int i = 0x1;
- **Function Parameter Qualifiers:** [none], in, out, inout
- **Global Variable Qualifiers:** const
- **Vector Components:** .xyzw .rgba .stpq
- **Flow Control:** if else for return break continue switch/case
- **Output:** vec4 fragColor
- **Input:** vec2 fragCoord
- **Preprocessor:** # #define #undef #if #ifdef #ifndef #else #elif #endif #error #pragma #line

Built-in Functions:

- type radians (type degrees)
- type degrees (type radians)
- type sin (type angle)
- type cos (type angle)
- type tan (type angle)
- type asin (type x)
- type acos (type x)
- type atan (type y, type x)
- type atan (type y_over_x)

- vec4 texture(sampler? , vec? coord [, float bias])
- vec4 textureLod(sampler, vec? coord, float lod)
- vec4 textureLodOffset(sampler? sampler, vec? coord, float lod, ivec? offset)
- vec4 textureGrad(sampler? , vec? coord, vec2 dPdx, vec2 dPdy)
- vec4 textureGradOffset sampler? , vec? coord, vec? dPdx, vec? dPdy, vec? offset)
- vec4 textureProj(sampler? , vec? coord [, float bias])
- vec4 textureProjLod(sampler? , vec? coord, float lod)
- vec4 textureProjLodOffset(sampler? , vec? coord, float lod, vec? offset)
- vec4 textureProjGrad(sampler? , vec? coord, vec2 dPdx, vec2 dPdy)

- type sinh (type x)
- type cosh (type x)
- type tanh (type x)
- type asinh (type x)
- type acosh (type x)
- type atanh (type x)

- type pow (type x, type y)
- type exp (type x)
- type log (type x)
- type exp2 (type x)
- type log2 (type x)
- type sqrt (type x)
- type inversesqrt (type x)

- type abs (type x)
- type sign (type x)
- type floor (type x)
- type ceil (type x)
- type trunc (type x)
- type fract (type x)
- type mod (type x, float y)
- type modf (type x, out type i)
- type min (type x, type y)
- type max (type x, type y)
- type clamp (type x, type minV, type maxV)
- type mix (type x, type y, type a)
- type step (type edge, type x)
- type smoothstep (type a, type b, type x)

- float length (type x)
- float distance (type p0, type p1)
- float dot (type x, type y)
- vec3 cross (vec3 x, vec3 y)
- type normalize (type x)
- type faceforward (type N, type I, type Nref)
- type reflect (type I, type N)
- type refract (type I, type N,float eta)

- vec4 texelFetch(sampler? , ivec? coord, int lod)
- vec4 texelFetchOffset(sampler?, ivec? coord, int lod, ivec? offset)
- vec? textureSize(sampler? , int lod)

- type dFdx (type x)
- type dFdy (type x)
- type fwidth (type p)

- type isnan (type x)
- type isinf (type x)
- float intBitsToFloat (int v)
- uint uintBitsToFloat (uint v)
- int floatBitsToInt (float v)
- uint floatBitsToUint (float v)
- uint packSnorm2x16 (vec2 v)
- uint packUnorm2x16 (vec2 v)
- vec2 unpackSnorm2x16 (uint p)
- vec2 unpackUnorm2x16 (uint p)

- bvec lessThan (type x, type y)
- bvec lessThanEqual (type x, type y)
- bvec greaterThan (type x, type y)
- bvec greaterThanEqual (type x, type y)
- bvec equal (type x, type y)
- bvec notEqual (type x, type y)
- bool any (bvec x)
- bool all (bvec x)
- bvec not (bvec x)

- float determinant(mat? m)
- mat?x? outerProduct(vec? c, vec? r)
- type matrixCompMult (type x, type y)
- type inverse (type inverse)
- type transpose (type inverse)

How-to

- **Use structs:** struct myDataType { float occlusion; vec3 color; }; myDataType myData = myDataType(0.7, vec3(1.0, 2.0, 3.0));
- **Initialize arrays:**float[] x = float[] (0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6);
- **Do conversions:** int a = 3; float b = float(a);
- **Do component swizzling:** vec4 a = vec4(1.0,2.0,3.0,4.0); vec4 b = a.zyyw;
- **Access matrix components:** mat4 m; m[1] = vec4(2.0); m[0][0] = 1.0; m[2][3] = 2.0;

Be careful!

- **the *f* suffix for floating pont numbers:** 1.0f is illegal in GLSL. You must use 1.0
- **saturate():** saturate(x) doesn't exist in GLSL. Use clamp(x,0.0,1.0) instead
- **pow/sqrt:** please don't feed sqrt() and pow() with negative numbers. Add an abs() or max(0.0,) to the argument
- **mod:** please don't do mod(x,0.0). This is undefined in some platforms
- **variables:** initialize your variables! Don't assume they'll be set to zero by default
- **functions:** don't call your functions the same as some of your variables

Shadertoy Inputs

vec3	**iResolution**	image/buffer	The viewport resolution (z is pixel aspect ratio, usually 1.0)
float	**iTime**	image/sound/buffer	Current time in seconds
float	**iTimeDelta**	image/buffer	Time it takes to render a frame, in seconds
int	**iFrame**	image/buffer	Current frame
float	**iFrameRate**	image/buffer	Number of frames rendered per second
float	**iChannelTime[4]**	image/buffer	Time for channel (if video or sound), in seconds
vec3	**iChannelResolution[4]**	image/buffer/sound	Input texture resolution for each channel
vec4	**iMouse**	image/buffer	xy = current pixel coords (if LMB is down). zw = click pixel
sampler2D	**iChannel{i}**	image/buffer/sound	Sampler for input textures i
vec4	**iDate**	image/buffer/sound	Year, month, day, time in seconds in .xyzw
float	**iSampleRate**	image/buffer/sound	The sound sample rate (typically 44100)

Shadertoy Outputs

Image shaders: fragColor is used as output channel. It is not, for now, mandatory but recommended to leave the alpha channel to 1.0.

Sound shaders: the mainSound() function returns a vec2 containing the left and right (stereo) sound channel wave data.