



Index

- 1D noise function octaves, 390–391
 - 1D shadow sampler, 203–204
 - 1D texture map, 481
 - 1D textures, 26, 28, 141, 203, 669
 - 1/f noise, 391
 - 2D noise, 394–395
 - 2D Perlin noise, 394
 - 2D shadow sampler, 204
 - 2D textures, 26, 141, 204, 268, 376, 669
 - glyphs, 272
 - pixel values, 28
 - 3D noise function, 395
 - 3D noise textures, 396–398
 - 3D textures, 2, 26–28, 669
 - 3Dlabs, 2, 56, 225, 226, 251, 505, 544
 - 3Dlabs developer Web site, 32, 63, 96, 165, 213, 230
 - 3Dlabs Web site, 63, 95, 208, 226, 230, 233, 257, 340, 396
 - 4D noise function, 395
- A**
- abs function, 128, 130, 458
 - Absent qualifier, 81
 - Absolute value, 137
 - Accumulation buffer, 7, 18, 669
 - acos function, 126, 530
 - Active attributes, 192–193, 598–600, 669
 - Active samplers, 204, 669
 - Active texture unit, 27, 31, 254, 258, 669
 - Active uniform variables, 201, 601–604, 669
 - Adaptive analytic
 - antialiasing fragment shader, 442
 - prefiltering, 440–443
 - ADD blend mode, 491
 - Addition operator (+), 85, 88
 - Additive operators, 94
 - Advanced RenderMan: Creating CGI for Motion Pictures* (Apodaca and Gritz), 147, 230, 323, 405, 444, 454
 - Aerial perspective, 525–529
 - Aggregate types, initializing, 75
 - AGL routines, 6
 - Airbrush and pen illustrations, 463
 - Akeley, Kurt, 4
 - Aliasing, 433–436, 443, 670
 - Aliasing artifacts, 164, 454
 - AlienRockArt*, 513
 - AlienRockArt* altgrad texture, 522–523
 - AlienRockArt* planet, 532
 - all function, 89, 141
 - all token, 92
 - Alpha, 670
 - Alpha test, 17, 670
 - Alpha value, 17
 - Altgrad map, 522–524
 - AlienRockArt*, 522–523
 - DragonRidges*, 523–524
 - snow, 522
 - Altitude angle, 268, 271
 - Ambient component, 368
 - Ambient light, 334
 - Ambient occlusion, 334–338, 342, 670





- Ambient variable, 289
- Amplitude, 427, 670
- Amplitude variable, 427
- Analytic integration, 443–446
- Ancient Chinese art of *chi ting*, 313
- Angle functions, 124–126
- Animatable for cloudy sky effect fragment shader, 413–414
- Animation, 411
 - blending factor, 414–415
 - fading object in or out, 417
 - key-frame interpolation, 414–416
 - linear blend, 414
 - morphing, 414–416
 - on/off, 412
 - particle systems, 418–426
 - threshold, 413
 - vertex noise, 417–418
 - wobble, 426–430
- Animation effects, 38, 411–413
- Anisotropic, 670
- Anisotropic glint, 365
- Anisotropic reflection, 369
- ANSI C programming language, 50
- Antialiased brick fragment shader, 446–447
- Antialiased checkerboard fragment shader, 448–449
- Antialiased stripe example, 437
 - adaptive analytic prefiltering, 440–443
 - analytic integration, 443–446
 - antialiased brick fragment shader, 446–447
 - generating stripes, 437–439
- Antialiasing, 16, 433, 454, 670
 - area sampling, 444
 - frequency clamping, 447–449
 - hatching example, 454
 - high-quality full-screen, 7
 - increasing resolution, 436–437
 - low-pass filtering, 439
 - user programmable methods, 38
 - value of integral over area of filter, 445
- any function, 89, 141
- API (application programming interface), 1, 670
- Apodaca, Tony, 323
- Apple, 2
- Application developers, 37
- ARB (Architecture Review Board), 49
- ARB prefix extensions, 3
- ARB_fragment_program extension, 56
- Arbitrary matrix, 23
- Arbitrary per-vertex data, 10
- ARB_vertex_program extension, 56
- Arc cosine, 126
- Arc sine, 126
- Arc tangent, 126
- Area lights, 312
- Area sampling, 444, 670
- Arithmetic operators, 88
- Array element operator (`[]`), 197
- Arrays, 72–74
 - indexing, 86
 - two-dimensional, 494
- Artifacts, 434
- asin function, 126
- Aspect ratio, 25
- Assembly language interfaces, 60
- Assignment operators (`+=`, `-=`, `*=`, and `/=`), 86, 89
- atan function, 126
- ATI, 2
- ATI developer Web site, 32, 231
- Atmosphere shaders, 543
- Atmospheric effects
 - aerial perspective, 525–529
 - RealWorldz*, 525–532
 - sky shading, 529–532
- atmospheric shell function, 526–528
- Attenuation, 237–238, 670
- Attribute aliasing, 189, 670
- Attribute qualified variables, 67, 78–79
- Attribute qualifiers, 78–79
- Attribute variables, 40–41, 51, 78, 80–81, 168, 671
 - built-in, 41–42, 97
 - data types, 186
 - initializing, 75
 - location of, 607–608
 - querying attribute binding for named vertex shader, 189–190
 - user-defined, 41–42, 97
 - vertex shader, 99
- Attributes, 79, 99–100
 - accessing in vertex shader, 97
 - locations, 185
 - three-dimensional, 22
 - vertex, 184–195
- Automatic mipmap generation, 2





Automatic state-tracking mechanism, 101
Auxiliary buffers, 7, 671
AVERAGE blend mode, 488
Azimuth angle, 268, 271

B

Back buffer, 6–7
Back-end processing and images, 20
Backface culling, 40
Back-facing polygons, 15
Background variable, 423
Bains, Inderaj, 233
Baker, Dan, 312
Baldwin, Dave, 149, 233, 444, 448, 468, 544
Barzel, Ronan, 322
Base color, 490
Base image, 488, 491–492
BEHIND blend mode, 488
Bent normal, 338, 671
bias parameter, 142
binormal variable, 191–192, 372
Binormal vector, 303
Bitmap rasterization, 479
Bitmaps, 18
Bitwise operators, 53, 94
Bit-wise shift operators, 94
Blend image, 491
Blend Modes, 487
Blend modes, 486–500
 ADD blend mode, 491
 AVERAGE blend mode, 488
 BEHIND blend mode, 488
 CLEAR blend mode, 489
 COLOR BURN blend mode, 490
 COLOR DODGE blend mode, 490
 CONVOLUTION blend mode, 493–495
 DARKEN blend mode, 489
 default, 488
 DIFFERENCE blend mode, 492
 DISSOLVE blend mode, 488
 edge detection, 498
 EXCLUSION blend mode, 492
 HARD LIGHT blend mode, 491
 INVERSE DIFFERENCE blend mode, 492
 LIGHTEN blend mode, 489
 MULTIPLY blend mode, 489
 NORMAL blend mode, 488
 OPACITY blend mode, 492–493
 OVERLAY blend mode, 490–491

 pixel-by-pixel blending, 487
 SCREEN blend mode, 490
 smoothing, 495–497
 SOFT LIGHT blend mode, 491
 SUBTRACT blend mode, 492
Blend variable, 416
Blending, 17
Blending functions, 2
Blinn, Jim, 313
bool data type, 52, 67–69
Boolean constructors, 77
Boolean values, 50, 69
Border color, 29
Border region, 512
Box filter, 443–444
Boyd, Chas, 312
BRDF (Bidirectional Reflectance Distribution Function), 368–375, 671
BRDF PTMs, 377–378
 application setup, 378–379
 light factor texture, 378–379
 stored as mipmap textures, 383
 surface rendering fragment shader, 382–383
 vertex shader, 379–381
break keyword, 82
Brick fragment shader, 162–163
Brick pattern, 149
 aliasing artifacts, 164
 color, 161–162
 depth, 164
 ease of modification, 158
 fragment shader, 157–164
 modeling coordinate position, 158
 mortar color, 161–162
 number, 159
 remaining constant, 156
 row number, 159
 size of bricks, 164
 vertex shaders, 151–157
Brick shaders
 application code, 208–212
 description of overall effect, 150
 overview, 150–151
Brick vertex shader, 157
BrickPct variable, 161
Brightness, 484
Buffer object, 12
Buffers, 17–18, 671
A Bug's Life, 544





- Built-in
 - attribute variables, 41–42, 97, 190
 - attributes, 79
 - constants, 98–99, 113–114
 - macros, 90
 - noise function, 399–400
 - sine functions, 84–85
 - texture functions, 249, 482
 - uniform variables, 42, 46, 51, 97, 105, 108–113, 196, 202
 - varying variables, 43, 97, 102–103, 105
 - Built-in functions, 52, 84–85, 123
 - angle functions, 124–126
 - common functions, 126, 128–134
 - exponential functions, 126–127
 - fragment processing functions, 144–145
 - geometric functions, 136–138
 - HLSL, 552
 - interesting effects in shaders, 124
 - matrix functions, 138–139
 - noise functions, 145–146
 - overloading, 52
 - overriding, 84–85
 - shader development, 219
 - texture access functions, 141–144
 - trigonometry functions, 124–126
 - usage, 124
 - variants, 123–124
 - vector relational functions, 139–141
 - vectors, 69
 - Built-in variables, 51, 67, 78, 81, 97, 101, 152, 546
 - global scope, 107
 - mapping OpenGL state values to, 113
 - vectors, 69
 - Bump function, 366
 - Bump mapping, 300, 671
 - application setup, 303–304
 - fragment shader, 306–308
 - increasing object realism, 301
 - light source, 301
 - location of bumps, 306
 - normal maps, 308–309
 - number of bumps per unit, 307
 - small effects to surfaces, 301
 - Bump mapping, *continued*
 - surface normal at each fragment location, 301
 - surface-local coordinate space, 302
 - tangent vector, 303–304
 - texture coordinates, 308
 - vertex shader, 306
 - width of bumps, 307
 - Bump maps, 308, 671
 - Bump pattern characteristics, 306–307
 - BumpDensity variable, 303, 306–307
 - BumpSize variable, 303, 307
 - “bumpy/shiner” shader pair, 268
 - Bunnell, Michael, 338, 343
 - bvec2 data type, 69
 - bvec3 data type, 69
 - bvec4 data type, 69
- ## C
- C functions
 - generating 3D noise texture, 396–397
 - installing brick shaders, 210–212
 - obtaining OpenGL and OpenGL Shading Language version information, 169–170
 - printing information log for object, 182–183
 - C programming language
 - drawing particles as points, 422
 - drawing silhouette edges on simple objects, 464
 - features not supported, 53
 - integers, 68
 - ISL (Interactive Shading Language) based on, 547
 - literal floating-point numbers, 68
 - updating time variable each frame, 423
 - vertex data for particles, 420–421
 - C++ programming language, 68
 - The C++ Programming Language* (Stroustrup), 95–96
 - The C Programming Language* (Kernighan and Ritchie), 95, 96
 - Calahan, Sharon, 323
 - Call by value-return, 54, 671
 - Calling conventions, 83–84
 - Cameras, 23
 - Caustic texture map, 536
 - CDs and diffraction grating, 365
 - ceil function, 128, 132
 - Ceramic effect, 271
 - Cerberus planet*, 524
 - Cg, 55, 544, 552–554
 - Cg Runtime library, 552–553
 - Cg shaders, 359, 552





- The Cg Tutorial* (Fernando and Kilgard), 359
- CgFX, 553
- Character buffer, 193
- Characters, 53
- Chromatic aberration, 362–363, 672
- Chromatic dispersion, 362, 672
- CIE (Committee Internationale de L'clairage), 482
- CIE system, 482–483
- clamp function, 129, 133–134, 435
- Clamping and fog factor, 246
- class keyword, 72
- CLEAR blend mode, 489
- Client-side memory, 12
- Client-side state, 8
- Clip space, 25, 672
- Clipping, 116–117, 672
 - point size, 115
 - primitives, 14–15
- Clipping coordinate system, 25, 672
- Clipping coordinates transformations, 234
- Clipping planes, 15, 25
- Cloud shader, 413–414
- Cloud texture, 260–261
- Cloud vertex shader, 401
- Clouds, 537–539
- Cohen, Elaine, 463
- ColAdjust variable, 272
- Color, 41
 - applying fog to compute final value, 246
 - to attach to fragment, 67
 - computing for vertex, 242
 - final surface computation, 242
 - final values with no lighting, 244
 - fragments, 157–158
 - mapping values, 481
 - new alpha value, 250
 - Pixel group, 19
 - primary, 242
 - replacing with value computed through texture access, 30
 - RGB representation of, 365
 - secondary, 242
 - terrain, 521–522
- Color buffers, 19, 593–595
- COLOR BURN blend mode, 490
- Color components, initializing, 18
- Color display, 5
- COLOR DODGE blend mode, 490
- Color index attribute, 100
- Color pickers, 158
- Color pixel rectangles, 18
- Color space conversions, 482–483
- Color sum, 16, 672
- Color values
 - accessing, 50
 - current array, 11
- color variable, 162
- Color vector, 69
- Commands
 - N in name, 186
 - out-of-order execution, 5
 - processing, 4
 - storing for later execution, 11
- Common coordinate system, 22
- Common math operations, 52
- Compaq, 1
- Compiled codes storage, 170
- Compiler front end, 225–226, 672
- Compiler/linker, 56–57
- Compilers, 54–55
 - ill-formed programs, 94–95
 - initial state, 92
 - optimization, 60
- Compile-time constants, 73, 80–81
- Compiling shader objects, 172–173, 581–582
- Complex number plane, 468–469, 472
- Complex numbers, 468–469
- Complex shapes, 7
- Components, 87–89
- Component-wise operation, 87–90
- Composite images, 7
- Compressed textures, 2, 28
- Computer Graphics Laboratory (Stanford), 544
- Confetti cannon vertex shader, 423–426
- const qualified variables, 80
- const qualifiers, 78, 83
- Constant qualified variables, 75
- constant qualifiers, 80–81
- Constants, 152
 - built-in, 113–114
 - compile-time, 80
- Constructive interference, 366
- Constructors, 52–53, 75–77, 672
- Containing tile, 512
- continue keyword, 82
- Contrast, 485
- Control constructs, 543





- Control texture, 287, 672
- Convolution, 443, 672
- CONVOLUTION blend mode, 493–495
- Convolution filter, 443, 493–495, 672
- Convolution kernel, 443, 493–495, 672
- Convolution operation
 - edge detection, 498
 - general fragment shader, 496–497
 - neighborhood averaging, 495–496
- Cook, Rob, 543
- coord texture coordinate, 143–144
- Coordinate spaces, 21, 546
- Coordinate system, 22
- Coordinate transforms, 21–26
- Coordinates and units of measurement, 22
- coord.s texture coordinate, 143
- coord.t texture coordinate, 143
- cos function, 124–125
- Cosine, 125
- CRC Standard Mathematical Tables and Formulas* (Zwillinger), 148
- createPoints subroutine, 420
- cross function, 136
- Cross product, 136
- Cube map textures, 2, 27–28, 247
- Cube mapping, 265, 673
- Cube mapping example, 265–268
- Cube maps, 26, 316, 672
 - accessing, 265–266
 - blending with stars, nebula, and atmosphere color, 530
 - built-in functions to access, 256
 - environment mapping, 267–268
 - faces, 266
 - OpenGL shaders, 265
- Culling, 15, 673
- currHdH variable, 520
- Curves, rendering with evaluators, 12
- D**
- DAG (directed acyclic graph), 227
- DARKEN blend mode, 489
- Data binding, 5
- Data structure, 55
- Data types
 - arguments linearly mapped to normalized range, 186
 - arrays, 73–74
 - automatic promotion, 53
 - conversing, 53
 - converting, 153
 - matrices, 70
 - OpenGL Shading Language, 546
 - passed by value, 84
 - promotion, 75
 - providing name, 68
 - RenderMan, 546
 - samplers, 71–72
 - scalars, 67–69
 - structures, 72
 - uniform qualified variables, 79
 - user-defined attribute variable, 192
 - vectors, 69–70
 - void, 74
- Daytime texture, 260, 263–264
- Debevec, Paul, 315–316, 322
- Debug pragma, 91
- Debugging shaders, 91, 220–222
- Decimal integer constant, 90
- Decrement operator (--), 88
- Default blend modes, 488
- Deferred shading, 673
 - soft shadows, 346–347
 - for volume shadows, 346–354
- #define directives, 90, 172
- defined operator, 90
- Degrees, converting radians to, 125
- degrees function, 125
- deLight program, 338, 340–341
- Dell, 2
- Delphi3D Web site, 32
- density variable, 245
- Dependent texture read, 46, 673
- Depth buffer, 7, 19, 673
- Depth comparison modes, 340
- Depth comparison values, 29
- Depth component textures, 340
- Depth components, 18
- Depth map, 339, 673
- Depth range, 40
- Depth test, 17, 673
- depth texture, 142
- Depth textures, 119, 253, 256
- Depth-cuing, 244, 673
- Depth-of-field effects, 7
- “The Design of the OpenGL Graphics Interface” (Segal and Akeley), 32
- The Design of the OpenGL Graphics Interface* (Segal and Akeley), 33
- Destination buffer, 18

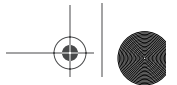




- Development aids, 206–207
- dFdx function, 145, 440–441, 456
- dFdy function, 145, 440–441, 456
- DIFFERENCE blend mode, 492
- Diffraction, 363–368, 673
- Diffraction gratings, 363–365, 673
- Diffraction vertex shader, 365
- Diffuse component, 368
- Diffuse environment maps, 317
- Diffuse lighting effects, 218
- Diffuse lighting factor, 264
- Diffuse reflection, 152–155, 317–319, 368
- DiffuseColor variable, 289
- diffuseContribution constant, 156
- Digital Equipment Corporation, 1
- Directional light sources, 13
- Directional lights, 236–237, 241
- DirectionalLight function, 243
- Directives, overriding earlier, 92
- DirectX 9, 549
 - .fx Effect format, 553
 - Software Development Kit, 544
- DirectX Effects Framework, 551
- disable behavior, 92–93
- discard keyword, 82, 106–108, 221, 300, 417
- Discontinuous jump at arbitrary point, 134
- Discontinuous stair-step pattern, 130, 132
- Disney, Walt, 411
- Disney Animation-The Illusion of Life* (Thomas and Johnston), 415
- Displacement shaders, 546
- Display, 5
- Display list mode, 11, 673
- Display lists, 11, 673
- Display memory, 5–6, 674
- Display of The Earth Taking Into Account Atmosphere Scattering* (Nishita, Shirai, Tadamura, and Nakamae), 525
- DISSOLVE blend mode, 488
- Distance, 136
- Distance attenuation, 116
- distance function, 136–137
- distanceShape, 327
- Distant objects, 525
- Dithering, 17
- Divide operator (/), 85
- Doss, Joshua, 233, 272
- dot function, 136
- Dot product, 136
- Dot product function, 481
- Dot product operation, 155, 360
- Double buffering, 6–7, 674
- Double-buffered windows, 7
- Double-precision floats, 53
- do-while statement, 82
- DragonRidges* altgrad map, 523–524
- DragonRidges* sky, 529–530
- Drawing
 - images, 18–21
 - immediate mode, 11
 - primitives, 9–11
- Drawing geometry
 - fragment processing, 16
 - frame buffer operations, 17–18
 - geometry specification, 9–12
 - per-fragment operations, 16–17
 - per-vertex operations, 12–14
 - primitive assembly, 14
 - primitive processing, 14–15
 - rasterization, 15–16
- drawPoints function, 422
- Driver model, 54–56
- Drivers, 54, 674
- Dynamic Ambient Occlusion and Indirect Lighting, 338
- E**
- Earth texture, 259
- ecPosition variable, 153–154
- Edge detection, 498
- Edge flag attribute, 100
- Effect nodes, 223
- Effect workspace, 223–224
- Effects, technical illustration example, 463
- Effects group, 223–224
- Effects group nodes, 223–224
- #elif directive, 90
- #else directive, 90
- else substatement, 74
- Embedded structures, 72
- Embossing effects, 491
- Empirical BRDF models, 370
- enable behavior, 93
- #endif directive, 90
- Entry points, 167
- Enumerated types, 53
- Environment mapping, 265, 267–268, 674
- Environment mapping example, 268–271
- Environment maps, 268, 316–317, 360
- Environments, tying OpenGL into, 6



- equal function, 89, 140
 - Equal operator (`==`), 89
 - Equality operators (`==` !=), 86
 - Equality operators (`==`, and `!=`), 89, 94
 - Equirectangular texture map, 268, 674
 - Error handling, 94–95
 - `#error` message directive, 90–91
 - Error messages, 91
 - Eta variable, 359
 - Evaluators, 12
 - Example shader pair, 65–67
 - EXCLUSION blend mode, 492
 - Executables, 36, 674
 - code storage, 170
 - installing, 56
 - Execution model, 4–5
 - `exp` function, 127
 - `exp` (base *e*) function, 245
 - `exp2` function, 127, 481
 - Explicit type conversions constructors, 77
 - Exponential functions, 126–127
 - Exponential operations, 52
 - Expressions, 93–94, 145
 - EXT prefix extensions, 3
 - `#extension` directive, 92
 - Extensions, 3–4, 92
 - Extrapolation, 483–486
 - Eye coordinate system, 23–24, 674
 - Eye coordinates, 23
 - light positions, 234–235
 - object position, 154
 - Eye space, 23–25, 235, 674
 - `EyeDir` variable, 270
 - `EyePosition` variable, 289
- F**
- faceforward function, 137
 - false literal Boolean constant, 68
 - `fBm`, 391
 - Fernando, Randima, 323
 - `__FILE__` macro, 90
 - Filter width, 444–445
 - Filtering, 435, 493–495, 674
 - Finding Nemo*, 323, 544
 - Fixed functionality, 9, 674
 - convolution operation, 493
 - disabling, 176
 - distance attenuation algorithm, 116
 - enabling texture on texture unit, 254
 - fragment processing, 39
 - graphics operations, 44–45
 - lighting model, 311
 - setting texture function for texture unit, 254
 - vertex processors, 39
 - Fixed-function processing, 47
 - Flat shading, 15, 674
 - float data type, 52, 67–69, 101, 106, 192
 - Floating-point matrix types, 51
 - Floating-point numbers, 70
 - Floating-point values, 50, 68, 113
 - Floating-point vector, 69
 - floor function, 128, 130, 132, 445–446
 - Flow control
 - built-in functions, 84–85
 - calling conventions, 83–84
 - functions, 82–83
 - Flow-control constructs, 69
 - `fltransform` function, 138
 - Focus point, 23
 - Fog, 16, 244–246, 675
 - Fog effect, 245
 - Fog factor, clamping, 246
 - For coordinate value, 103
 - for statement, 74, 82
 - Formal function parameters, 78
 - `fract` function, 128, 132, 159, 161, 438, 446, 510
 - Fractal terrains, 508–509
 - noise functions, 510
 - surface normals, 513
 - Fractals, 675
 - Fractional Brownian motion, 391
 - Fragment processing, 2, 16, 35–36, 675
 - fixed functionality, 39
 - functions, 144–145
 - Fragment processor, 38–39, 42–47, 675
 - fragment processing functions, 144
 - programmable, 39
 - special input variables, 106
 - special output variables, 107–108
 - uniform variables, 105–106
 - varying variables, 104–105
 - Fragment shaders, 36, 44, 258, 675
 - accessing current state, 42
 - accessing texture, 255–256
 - adaptive analytic antialiasing, 442
 - adding together two images, 487
 - animatable for cloudy sky effect, 413–414
 - animation effects, 426
 - antialiased brick, 446–447
 - antialiased checkerboard, 448–449



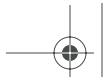
- attributes, 79
- bias parameter, 142
- BRDF PTM surface rendering, 382–383
- brick pattern, 157–164
- brightness, 484
- built-in constants, 113
- built-in varying variables, 97
- bump mapping, 306–308
- chromatic aberration effect, 363
- cloudy sky effect, 402
- color, depth, and arbitrary values, 46
- color value, 107
- common functions, 126, 128–134
- communicating results to processing stages, 97
- contrast, 485
- cube mapping example, 267–268
- defining values passed on to, 153
- discard keyword, 221, 300
- empty shader objects, 209
- environment mapping example, 269–271
- executing in parallel, 513
- exponential functions, 126–127
- fog factor, 246
- Fresnel reflection/refraction effect, 361–362
- general convolution computation, 497
- generating shadows, 343–346
- geometric functions, 136–138
- glyph bombing, 277–281
- Gooch shading, 466–467
- granite, 405
- image-based lighting, 318, 336–337
- infrequently changing values to, 46
- input variables, 97
- lighting computation, 153
- lighting operation, 301
- main function, 82
- Mandelbrot example, 472–474
- marble, 404
- matrix functions, 138–139
- maximum number of texture image units, 119
- mitigating aliasing effects, 437
- multiple values for fragments, 204
- multitexturing example, 263–265
- neighborhood averaging convolution, 496
- noise, 401–402
- noise functions, 145–146
- obtaining input data, 105
- OpenGL Shading Language, 546
- output, 107
- output and debugging, 221–222
- perturbation calculation, 427
- perturbation factor, 427–428
- procedurally discarding part of object, 300
- procedurally generated wood, 406–408
- process texture values, 480
- processing fragments, 480
- random offsets, 279
- rapidly changing characteristics, 218
- reading
 - multiple values from texture, 46
 - varying qualified shader, 80
 - varying variables, 102
- refracted vector, 358
- rendering with BRDF model, 374
- results from preceding processing, 97
- saturation, 485
- shader computation, 218
- shadow maps, 343–346
- sharpness, 486
- sky color texture map lookup, 530–531
- soft volume shadow algorithm, 350, 353
- special built-in fragment shader input variables, 97
- special input variables, 104
- specialized functions, 52
- spherical harmonics lighting, 322
- state, 46, 97
- subset of varying variables, 43
- sun surface, 403
- supersampling, 436
- supporting parallelism argument-processing level, 44
- technical illustration example, 466–467
- texture access functions, 141–144
- texture maps, 104
- texturing, 118–120, 259
- tile sets, 518–519
- time to generate texture map, 525
- toy ball, 294–299
- trigonometry functions, 124–126
- überlight model, 326–329
- uniform variables, 104
- unsharp masking, 499–500
- varying variables, 104, 163
- vector relational functions, 139–141
- vertex shader communications, 79–80
- visual feedback about, 221–222
- wobble effect, 429–430
- woodcut-style rendering, 461–462





- Fragments, 15, 17, 35, 675
 - color, 157–158
 - color computing, 51
 - color to attach to, 67
 - creation of, 66
 - depth computing, 51
 - depth value, 107, 157–158
 - discarding, 108
 - facing direction, 106
 - front facing, 106
 - generated by rasterizing front-facing primitive, 45
 - generating, 20
 - location within current brick, 160
 - modeling coordinates, 159
 - parallel processing, 39
 - position within stripe pattern, 290
 - preventing from updating frame buffer, 82
 - RGB values, 250
 - window coordinate position, 45
 - Frame buffers, 5–7, 8
 - auxiliary buffers, 7
 - buffers, 17–18
 - characteristics for windows, 6
 - drawing graphics into, 4
 - initializing values, 18
 - nonwindowed system, 6
 - operations, 17–18
 - preventing update, 221
 - reading back values stored in, 4
 - updating single location in, 35
 - windowing system, 6
 - Framebuffer operations, 675
 - Freq uniform variable, 428
 - Freq variable, 427
 - Frequency, 427, 675
 - Frequency clamping, 447–449
 - Fresnel approximation equation, 360
 - Fresnel effect, 358, 675
 - Fresnel equations, 358–359
 - Fresnel reflection/refraction effect, 361–362
 - FresnelPower, 360
 - Freudenberg, Bert, 437, 454, 460–461
 - Front buffer, 6–7
 - Front-facing polygons, 15
 - Front-facing primitive, 106
 - Frustum, 40, 675
 - Frustum clipping, 25, 676
 - ftransform function, 118, 136, 157, 349, 416
 - Full OpenGL Pipeline shader, 233
 - Full-screen antialiasing, 7
 - Function tree, 519–521
 - Functions, 82–83
 - accessing values in texture memory, 52
 - actual value, 447
 - aliasing variables within, 54
 - average value, 447
 - avoiding calls to expensive, 520
 - built-in, 36, 52, 84–85, 123
 - calling, 82
 - calling by value-return, 36
 - calling conventions, 83–84
 - convolution, 443
 - declaring, 52
 - definition (body) or declaration must be in scope, 82
 - empty parameter list, 82–83
 - encoding complex with textures, 220
 - existing from, 83
 - gradient, 441
 - input parameters, 54
 - lacunarity, 393
 - never producing negative values, 130
 - noise values for procedural texturing effects, 52
 - output parameters, 54
 - overloading, 36, 52, 82, 123
 - query functions, 178–184
 - recursively calling, 83
 - returning no value, 74
 - returning value or returning nothing, 84
 - structures and arrays passed as arguments, 84
 - void type, 84
 - when to copy parameters, 83
 - Fuzz variable, 290
 - Fuzzing transition region, 291
 - fwidth function, 145, 441, 444–445
 - .fx Effect format, 553
- ## G
- Gaussian filter, 496–497
 - Gaussian reflectance model, 371
 - General computation, 38
 - General convolution shader, 496–497
 - Generic attribute index, 189
 - Generic vertex attribute array
 - defining, 657–659
 - enabling or disabling, 188, 596–597





- Generic vertex attributes, 41–42, 186
 - associating index with named attribute variables, 578–580
 - explicitly binding to attribute variable, 188–189
 - parameters, 194–195, 624–626
 - querying state, 194–195
 - value of, 651–656
 - vertex array pointer for, 187
- Geometric functions, 136–138
- Geometric image transforms, 480
- Geometric operations, 52
- Geometric primitives, 9, 676
- Geometry
 - drawing with vertex arrays, 99
 - sending data to OpenGL, 9–12
- Geometry specification, 9–12
- getUniLoc function, 210
- glAccum, 18
- GL_ACTIVE_ATTRIBUTES parameter, 180
- GL_ACTIVE_ATTRIBUTE_MAX_LENGTH parameter, 180
- glActiveTexture, 27, 254
- GL_ACTIVE_UNIFORM_MAX_LENGTH parameter, 180
- GL_ACTIVE_UNIFORMS parameter, 180
- GL_ADD symbolic constant, 249–250
- Glaeser, Georg, 532
- glAlphaFunc, 17
- Glanville, Steve, 272
- GL_ATTACHED_SHADERS parameter, 180
- glAttachObjectARB function, 660
- glAttachShader function, 55, 57, 167, 173–174, 576–577, 660
- GL_AUX0 symbolic constant, 351
- GL_AUXi symbolic constant, 205
- gl_BackColor constant, 105
- gl_BackColor variable, 103, 114–115, 117, 243
- GL_BACK_LEFT symbolic constant, 205, 349
- glBackMaterial.shininess value, 243
- GL_BACK_RIGHT symbolic constant, 205
- gl_BackSecondaryColor variable, 103, 105, 114–115, 117, 243
- glBegin function, 9–10, 14
- glBindAttribLocation function, 42, 57, 168, 188–189, 191, 303, 372, 578–580, 660
- glBindAttribLocationARB function, 660
- glBindBuffer command, 12
- glBindTexture, 28, 254
- glBitmap, 18, 480
- GL_BLEND symbolic constant, 249–250, 351
- glBlendColor, 17
- glBlendEquation, 17
- glBlendFunc, 17, 351
- glBufferData command, 12
- glBufferSubData command, 12
- glCallList function, 11
- glCallLists function, 11
- glClear, 18
- glClearAccum, 18
- glClearColor, 18
- glClearDepth, 18
- glClearStencil, 18
- glClientActiveTexture, 30
- glClipPlane function, 15, 25, 116
- gl_ClipVertex variable, 43, 102, 116
- glColor function, 10, 13, 99
- gl_Color variable, 41, 100, 103–105, 184, 423
- glColorMask function, 18
- glColorPointer function, 11–12
- GL_COMBINED_TEXTURE_IMAGE_UNITS constant, 204
- glCompileShader function, 55, 57, 167, 172, 209, 581–582, 660
- glCompileShaderARB function, 660
- GL_COMPILE_STATUS parameter, 178
- glCompressedTexImage1D/2D/3D, 28
- glCompressedTexSubImage1D/2D/3D, 28
- glCopyPixels, 19–20
- glCopyTexImage, 19–20
- glCopyTexImage1D/2D, 28
- glCopyTexImage2D, 351
- glCopyTexSubImage, 19–20
- glCopyTexSubImage1D/2D/3D, 28
- glCreateProgram function, 57, 167, 173, 583–584, 660
- glCreateProgramObjectARB function, 660
- glCreateShader function, 54, 57, 167, 171, 585–586, 660
- glCreateShaderObjectARB function, 660
- glCullFace, 15
- GL_CURRENT_COLOR symbolic constant, 8
- GL_CURRENT_PROGRAM symbolic constant, 183, 575
- GL_CURRENT_PROGRAM symbolic constant, 661
- GL_CURRENT_VERTEX_ATTRIB parameter, 195
- GL_DECAL symbolic constant, 249–250





- glDeleteObjectARB function, 660
- glDeleteProgram function, 57, 177, 587–588, 660
- glDeleteShader function, 58, 177, 589–590, 660
- GL_DELETE_STATUS parameter, 178–179
- GL_DEPTH_COMPONENT, 119, 256
- glDepthFunc, 17
- glDepthRange, 15, 26
- glDetachObjectARB function, 660
- glDetachShader function, 58, 177, 591–592, 660
- glDisable, 115 function, 235
- glDisable function, 8
- glDisableClientState function, 8
- glDisableVertexAttribArray function, 58, 188, 661
- glDisableVertexAttribArrayARB function, 661
- glDrawArrays function, 11, 99, 188
- glDrawBuffer, 18
- glDrawBuffers function, 18, 108, 205–206, 349, 593–595, 664
- glDrawBuffersARB function, 664
- glDrawElements function, 11
- glDrawPixels, 18–19, 28, 480, 487
- glDrawRangeElements function, 11
- glEnable function, 8, 15–16, 25, 115, 235, 254, 351
- glEnableClientState function, 8, 422
- glEnable/glDisable function, 13, 27
- glEnableVertexArrayPointer, 168
- glEnableVertexAttribArray function, 58, 188, 422, 596–597, 661
- glEnableVertexAttribArrayARB function, 661
- glEnd function, 10
- glEndList function, 11
- GL_EXP symbolic constant, 245
- GL_EXP2 symbolic constant, 245
- GL_EXTENSIONS symbolic constant, 3, 20
- GL_EYE_LINEAR symbolic constant, 246, 248
- glFinish, 18
- GL_FLOAT symbolic constant, 193
- GL_FLOAT_MAT2 symbolic constant, 193
- GL_FLOAT_MAT3 symbolic constant, 193
- GL_FLOAT_MAT4 symbolic constant, 193
- GL_FLOAT_VEC2 symbolic constant, 193
- GL_FLOAT_VEC3 symbolic constant, 193
- GL_FLOAT_VEC4 symbolic constant, 193
- glFlush, 18
- glFog, 16, 103
- gl_fog.color uniform variable, 51
- GL_FOG_COORDINATE symbolic constant, 103, 105
- gl_Fog.density variable, 245
- gl_Fog.end variable, 245
- gl_FogFragCoord variable, 103, 117, 245
- gl_Fog.start variable, 245
- gl_FragColor variable, 46, 51, 67, 78, 107–108, 160–163, 402
- gl_FragCoord variable, 45, 106–107
- gl_FragData array, 204
- gl_FragData variable, 46, 107–108
- gl_FragDepth variable, 46, 51, 107–108
- GL_FRAGMENT_DEPTH symbolic constant, 103, 105
- GL_FRAGMENT_SHADER, 171
- gl_FrontColor variable, 103, 105, 114–115, 117, 242, 314, 365–366
- glFrontFace, 16, 106
- gl_FrontFacing variable, 45, 106
- GL_FRONT_LEFT symbolic constant, 205
- gl_FrontLightModelProduct.sceneColor variable, 242
- glFrontMaterial.shininess value, 243
- GL_FRONT_RIGHT symbolic constant, 205
- gl_FrontSecondaryColor variable, 103, 105, 114–115, 117, 242
- glFrustum function, 25
- glGet function, 8, 13, 27, 98, 100–101, 103, 106, 113–114, 118, 183, 203, 661
- glGetActiveAttrib function, 58, 192–193, 598–600, 661
- glGetActiveAttribARB function, 661
- glGetActiveUniform function, 58, 201–203, 601–604, 661
- glGetActiveUniformARB function, 661
- glGetAttachedObjectsARB function, 661
- glGetAttachedShaders function, 58, 183, 605–606, 661
- glGetAttribLocation function, 58, 168, 189–190, 607–608, 661
- glGetAttribLocationARB function, 661
- glGetclipPlane function, 8
- glGetError function, 8
- glGetHandleARB function, 661
- glGetInfoLogARB function, 661
- glGetLight function, 8
- glGetMaterial function, 8
- glGetObjectParameterARB function, 663
- glGetObjectParameterfvARB function, 663





- glGetProgram function, 58, 177, 193, 609–611, 663
- glGetProgramInfoLog function, 58, 175, 612–613, 661
- glGetProgramiv function, 179–180
- glGetShader function, 58, 172, 177, 209, 614–615, 663
- glGetShaderInfoLog function, 58, 172, 181–182, 616–617, 661
- glGetShaderiv function, 178–179
- glGetShaderSource function, 58, 180–181, 618–619, 663
- glGetShaderSourceARB function, 663
- glGetString function, 3, 20, 168–169
- glGetUniform function, 58, 197, 199, 620–621, 663
- glGetUniformARB function, 663
- glGetUniformfv function, 199–200
- glGetUniformiv function, 199–200
- glGetUniformLocation function, 42, 58, 168, 196–197, 199–200, 622–623, 663
- glGetUniformLocationARB function, 663
- glGetVertexAttrib function, 58, 624–626, 663
- glGetVertexAttribARB function, 663
- glGetVertexAttribdv function, 194
- glGetVertexAttribfv function, 194
- glGetVertexAttribiv function, 194
- glGetVertexAttribPointer function, 58, 627–628, 663
- glGetVertexAttribPointerARB function, 663
- glGetVertexAttribPointerv function, 195
- GL_INFO_LOG_LENGTH parameter, 179–180
- glInterleavedArrays function, 11
- GL_EQUAL, 464
- GL_LESS, 464
- glLight function, 13, 24
- GL_LIGHTING symbolic constant, 13
- glLightModel function, 13–14
- gl_LightSource[i] uniform variable, 51
- GL_LINEAR symbolic constant, 244, 482
- GL_LINE_SMOOTH symbolic constant, 16
- glLineStipple, 16
- glLineWidth, 16
- GL_LINE_WIDTH symbolic constant, 8
- glLinkProgram function, 55, 58, 167, 189, 210, 422, 632–635, 663
- glLinkProgramARB function, 663
- GL_LINK_STATUS parameter, 179
- glLoadIdentity, 23
- glLoadMatrix function, 13, 23
- glLogicOp, 17
- glIsProgram function, 58, 184, 629–630, 664
- glIsShader function, 58, 184, 631, 664
- glMapBuffer function, 12
- glMaterial, 13
- glMatrixMode function, 13, 25, 31
- GL_MAX_COMBINED_TEXTURE_IMAGE_UNITS symbolic constant, 119, 207, 574
- GL_MAX_DRAW_BUFFERS symbolic constant, 108, 207, 574
- GL_MAX_FRAGMENT_UNIFORM_COMPONENTS symbolic constant, 106, 207, 574
- gl_MaxLights, 114
- GL_MAX_LIGHTS symbolic constant, 13
- GL_MAX_TEXTURE_COORDS symbolic constant, 103, 119, 207, 574
- GL_MAX_TEXTURE_IMAGE_UNITS symbolic constant, 119, 203–204, 207, 574
- GL_MAX_TEXTURE_UNITS symbolic constant, 118–119
- GL_MAX_VARYING_FLOATS symbolic constant, 103, 207, 574
- GL_MAX_VERTEX_ATTRIBS symbolic constant, 100, 207, 574
- GL_MAX_VERTEX_TEXTURE_IMAGE_UNITS symbolic constant, 119, 203–204, 207, 574
- GL_MAX_VERTEX_UNIFORM_COMPONENTS symbolic constant, 101, 207, 575
- gl_ModelViewMatrix variable, 51, 100, 152–153
- gl_ModelViewProjectionMatrix variable, 152
- GL_MODULATE symbolic constant, 249
- glMultiDrawArrays function, 11
- glMultiDrawElements function, 11
- glMultiTexCoord function, 30
- gl_MultiTexCoord0 variable, 100, 184
- gl_MultiTexCoord1 variable, 100, 184
- gl_MultiTexCoord2 variable, 100, 184
- glMultMatrix function, 13
- GL_NEAREST, 482
- glNewList function, 11
- GL_NONE symbolic constant, 205
- glNormal function, 10, 41, 99
- gl_Normal variable, 41, 79, 100, 152, 154, 184, 372, 416
- GL_NORMALIZE symbolic constant, 235
- GL_NORMAL_MAP symbolic constant, 247
- gl_NormalMatrix variable, 152, 154
- glNormalPointer function, 11–12
- gl_NormalScale variable, 235





- Global lighting parameters, 14
Global variables and initializers, 82
GL_OBJECT_LINEAR symbolic constant, 246, 248
GL_ONE, 348, 351
GL_ONE_MINUS_DEST_ALPHA, GL_DST_ALPHA blend mode, 537
glOrtho function, 25
Gloss maps, 260, 262, 676
Gloss texture, 260
glPixelMap function, 19
glPixelStore, 19, 21
glPixelTransfer, 19
glPixelZoom, 20
glPointParameter, 16
glPointSize function, 16, 115–116
gl_PointSize variable, 43, 101, 115
GL_POINT_SMOOTH symbolic constant, 16
glPolygonMode, 16
glPolygonOffset function, 16, 339
GL_POLYGON_SMOOTH symbolic constant, 16
glPolygonStipple, 16
glPopAttrib function, 8
glPopClientAttrib function, 8
glPopMatrix, 24
gl_Position variable, 43, 51, 67, 78, 101, 117, 138, 153, 163, 365–366, 400, 424, 455
glPushAttrib function, 8
glPushClientAttrib function, 8
glPushMatrix, 24
glRasterPos function, 20, 117
glReadBuffer function, 20, 351
glReadPixels, 19–20
GL_REFLECTION_MAP symbolic constant, 247
GL_REPLACE symbolic constant, 249
glRotate function, 13, 24
glScale function, 13, 24
glSecondaryColor function, 13
glSecondaryColor function, 16
gl_SecondaryColor variable, 100, 103–105, 184
glShadeModel function, 15
glShaderSource function, 54, 58, 167, 171, 209, 636–637, 663
glShaderSourceARB function, 663
GL_SHADER_SOURCE_LENGTH parameter, 179
GL_SHADER_TYPE parameter, 171, 178
GL_SHADING_LANGUAGE_VERSION symbolic constant, 169, 575
GLSL. *See* OpenGL Shading Language
GLSL demo, 396
GLSL validate, 226
GLSLdemo file, 257
GLSLParserTest, 226
GL_SPHERE_MAP symbolic constant, 247
GL_SRC_ALPHA, 348, 351
glStencilFunc, 17
glStencilMask function, 18
glStencilOp, 17
gl_TexCoord[] array, 103, 105, 117
glTexCoord function, 10, 30, 99
gl_TexCoord[0] variable, 258
glTexCoordPointer, 30
glTexEnv function, 27, 30–31, 254
glTexGen function, 27, 30
glTexImage function, 18–20, 27, 254
glTexImage1D function, 28
glTexImage2D function, 28, 257
glTexImage3D function, 28, 398
glTexParameter function, 27, 29, 254, 256–257
glTexSubImage function, 18–20
glTexSubImage1D/2D/3D functions, 28
GL_TEXTURE_1D, 119
GL_TEXTURE_2D, 119
GL_TEXTURE_3D, 119
GL_TEXTURE_CUBE_MAP, 119
gl_TextureMatrix variable, 235
gl_TextureMatrix[1] variable, 342
glTranslate function, 13, 24
gluLookAt function, 24
glUniform function, 42, 59, 168, 197, 199–200, 638–644, 663
glUniform function, 197
glUniformv function, 198
glUniform1i data type, 198, 203
glUniform1iv data type, 198, 203
glUniformARB function, 663
glUniformMatrixfv function, 198
glUnmapBuffer function, 12
gluPerspective function, 25
glUseProgram function, 56, 59, 167, 176, 184, 210, 645–648, 664
glUseProgramObjectARB function, 664
glValidateProgram function, 59, 206, 649–650, 664
glValidateProgramARB function, 664
GL_VALIDATE_STATUS symbolic constant, 180
GL_VERSION symbolic constant, 169
glVertex function, 10, 41, 100, 184, 186





- gl_Vertex variable, 41, 79, 100, 138, 152–153, 156, 184, 192, 372, 381, 416, 423
- glVertex2 function, 187
- glVertex3 function, 187
- glVertex4 function, 187
- glVertexAttrib function, 42, 59, 100, 168, 185–187, 190, 372, 651–656, 664
- glVertexAttrib4 function, 185–186
- glVertexAttrib4N function, 186
- glVertexAttrib4Nub function, 186
- glVertexAttribARB function, 664
- GL_VERTEX_ATTRIB_ARRAY_ENABLED parameter, 194
- GL_VERTEX_ATTRIB_ARRAY_NORMALIZED parameter, 195
- GL_VERTEX_ATTRIB_ARRAY_SIZE parameter, 194
- GL_VERTEX_ATTRIB_ARRAY_STRIDE parameter, 194
- GL_VERTEX_ATTRIB_ARRAY_TYPE parameter, 195
- glVertexAttribPointer function, 59, 168, 187, 657–659, 664
- glVertexAttribPointerARB function, 664
- glVertexPointer function, 11–12
- GL_VERTEX_PROGRAM_POINT_SIZE symbolic constant, 115
- GL_VERTEX_PROGRAM_TWO_SIDE symbolic constant, 115
- GL_VERTEX_SHADER, 171
- glViewport, 15, 26
- glWindowPos, 20
- GLX routines, 6
- Glyph bombing, 676
 - application setup, 272–276
 - diffuse factor, 276
 - fragment shader, 277–281
 - looping, 275, 277–278
 - random number texture, 278
 - scaled texture coordinates, 277
 - vertex shader, 276–277
- Glyphs
 - corresponding rotation matrix, 278–279
 - placing randomly, 272
 - probability of drawing in cell, 274–275
 - random angle of rotation, 274, 278
 - random color, 272
 - random offsets, 273–274
 - scaling size, 274
 - smoothly antialiased edge, 279
- Gooch, Amy, 463
- Gooch, Bruce, 463
- Gooch shader, 466
- Gooch shading, 676
 - automatically generating important edges, 463–464
 - black curves representing important edges, 463
 - color ramp, 465
 - computing colors, 465
 - diffuse reflection factor, 465
 - drawing silhouette edges for simple objects, 464
 - edges rendered as antialiased black lines, 463
 - fragment shader, 466–467
 - luminance values, 465
 - luminance values conveying information
 - about curvature, 464–465
 - shading of object, 465
 - specular highlights, 464
 - warm-to-cool color gradient, 465
- Gooch vertex shader, 466
- goto keyword, 82
- Gourad shading, 676
- GPU Gems* (Glanville), 272, 323, 343, 364
- GPU Gems 2*, 338
- Gradient noise, 393, 676
- Gradient vectors, 440–441, 676
- Gradients, 441, 457, 676
- GrainScale variable, 408
- Grammar
 - OpenGL Shading Language in terms of tokens, 559–571
 - output of lexical analysis tokens, 559
- Gram-Schmidt orthonormalization, 381
- Granite, 404–405
- Granite fragment shader, 405
- Graphics accelerators, 5, 676
- Graphics contexts, 6, 8, 676
- Graphics hardware
 - based on programmable processor technology, 49
 - capabilities, 2
 - double buffering, 6–7
 - exposing programmability, 48
 - parallel processing, 49
 - programmability, 4
 - RenderMan, 546
 - single-precision floating-point calculations, 520





Graphics primitives, 18, 25
Graphics processing pipeline, 8–9, 676
Grayscale display, 5
Grayscale pixel rectangles, 18
greaterThan function, 140
greaterThanEqual function, 140
Gritz, Larry, 323, 448
Gruschel, Jens, 487

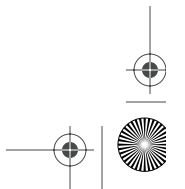
H

Haeberli, Paul, 483
Hanrahan, Pat, 318–319
HARD LIGHT blend mode, 491
Hardware
 exposing functionality, 123
 exposing new functionality, 3
 graphics accelerators, 5
 optimal performance on wider range, 60
 vector-processing capabilities, 69
Hardware independence, 2
hardware independence, 48
Hash-based operators (#, ##, etc.), 94
Hatching example
 adding character, 459–460
 antialiasing, 454
 application setup, 455
 density too high, 457
 generating hatching strokes, 456
 hatching fragment shader, 461–462
 imperfections, 459–460
 noise added to stripe pattern, 460
 simulating lighting, 458–459
 softening edge, 458
 uniform line density, 456–458
 vertex shader, 455–456
Hatching fragment shader, 461–462
Hatching shader, 454–455
HDR (high dynamic range) images, 316, 320–321
HDRShop, 316–317
HDTV standard, 483
Hemisphere lighting, 311–313, 676
 ambient occlusion, 335–336
 ground color, 314
 sky color, 314
 vertex shader, 314–315
Heterofractals, 508–509
Hidden-surface removal, 7
HIL (high-level intermediate language), 225

HLSL (High-Level Shader Language), 55, 223, 544, 549–552
HLSL pixel shader, 551
HLSL shaders, problems with, 551
HLSL vertex shaders, 550
Homogeneous vertex position, 157

I

IBM, 1–2
IDE (integrated development environment), 223
Identity matrix, 23
#if directive, 90
if statement, 74, 82
#ifdef directive, 90
if-else statement, 82
#ifndef directive, 90
Ill-formed programs, 94–95
Image processing, 38
An Image Synthesizer (Perlin), 148
Image-based lighting, 315–318, 336–337, 677
Imager shaders, 546
Images
 averaging, 488
 back-end processing, 20
 blend modes, 486–500
 brightening base color, 490
 brightness, 484
 combining blend image with base image, 491
 contrast, 485
 darkening, 489–490
 difference between blend and base image, 492
 drawing, 18–21
 edge detection, 498
 extrapolation, 483–486
 filtering, 493–495
 hard light, 491
 interpolation, 483–486
 lightening, 489
 luminance, 490–491
 multiplying inverse, 490
 multiplying values, 489
 muting, 491
 opacity, 492–493
 pixel transfer, 19–20
 pixel unpacking, 19
 randomness, 488
 raster position, 20



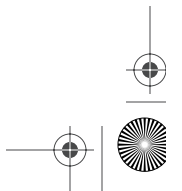


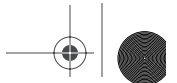
- rasterization, 20
 - read control, 20–21
 - saturation, 485
 - sharpening, 499–500
 - sharpness, 486
 - smoothing, 495–497
 - soft contrast, 492–493
 - soft light, 491
 - subtracting blend image from base image, 492
 - for texture memory, 18
 - vanishing point, 25
 - zoom factor, 20
 - Imaging, 479
 - color space conversions, 482–483
 - geometric image transforms, 480
 - lookup table operations, 481–482
 - mathematical mappings, 481
 - multiple pixel access, 480
 - operations, 479
 - scale-and-bias, 481
 - single pixel access, 480
 - Imaging subset, 19–20, 677
 - Immediate mode, 11, 677
 - Implementation-dependent
 - API values, 574–575
 - constants, 113–114
 - values, 113, 207
 - Improving Noise* (Perlin), 148
 - in qualifier, 54, 83
 - Incident vector, 154
 - #include directives, 53
 - Increment operator (++), 88
 - Index of refraction, 358, 677
 - Index ([]) operator, 85
 - Indexing, 86
 - Industrial Light & Magic, 338
 - Information log, 172–173, 175
 - error messages, 91
 - length of, 182
 - program objects, 182, 210, 612–613
 - reviewing, 220
 - shader objects, 181, 616–617
 - shaders, 209
 - init2Dtexture function, 262
 - Initializers, 75–77
 - brace syntax “/,” 75
 - const qualified variables, 80
 - constructors, 52
 - global variables, 82
 - inorout counter, 295
 - inout qualifier, 54, 83
 - Input function box filter, 443–444
 - int data type, 67–69
 - Integer, 128
 - Integers, 132
 - precision, 68
 - vector types, 50
 - Integrating the OpenGL Shading Language* (Lichtenbelt), 120–121
 - Intel, 1–2
 - Interactive Rendering with Arbitrary BRDFs Using Separable Approximations* (Kautz and McCool), 381
 - Interior region, 512
 - Interpolated varying variables, 45
 - Interpolation, 483–486
 - INVERSE DIFFERENCE blend mode, 492
 - inversesqrt function, 127
 - Iris GL (Graphics Library), 1–2
 - ISL (Interactive Shading Language), 547–549
 - ISL shaders, 547–549
 - Isotropic, 677
 - ITU-R Recommendation BT.709 (HDTV color standard), 481, 483
 - ivec2 data type, 69
 - ivec3 data type, 69
 - ivec4 data type, 69
- J**
- Johnston, Scott, 454
 - Julia sets, 474–475
 - Jurassic Park*, 544
 - Jurassic Park III*, 338
- K**
- Kd variable, 289
 - Kessenich, John, 149, 268, 304, 396
 - Key-frame interpolation, 414–416, 677
 - Koren, Steve, 149, 468
- L**
- Labels, 82
 - Lacunarity, 393, 677
 - lambda2rgb function, 365–366
 - Lander, Jeff, 464
 - Lander technique, 466





- Landis, Hayden, 338
- Laplacian operator, 498
- Latitude-longitude texture map, 677
- Lat-long environment map, 316
- Lat-long texture map, 268
- Lattice, 299–300
- Lawrence Berkeley Laboratory, 370
- LCcamera variable, 325
- LCnorm variable, 325
- LCpos variable, 325
- length function, 136–137, 237, 457
- lessThan function, 139
- lessThanEqual function, 89, 140
- Level-of-detail, 142, 677
 - argument, 28
 - bias, 256
 - clamping and biasing value, 29
- Lexical analysis, 225, 677
- Licea-Kane, Bill, 292
- Lichtenbelt, Barthold, 208, 257
- Light
 - attenuation factor, 326–327
 - behavior of visible, 364
 - diffraction, 363–368
 - intensity interpolated value, 158
 - reflection, 155, 327
 - refraction, 358–363
 - separating based on wavelength, 364
 - shape and placement, 323
 - soft edge, 324
 - surface-local, 306
 - transforming, 302
 - wavelengths, 364
- Light coordinate system, 326
- Light direction vector, 154–155
- Light factor texture, 378–379
- Light positions
 - eye coordinates, 234–235
 - texture unit, 258
 - transforming by modelview matrix, 234–235
 - wobbles, 427
- Light probe, 315, 678
- Light probe gallery, 319
- Light probe images, 315–316, 318–319, 322, 678
- Light shaders, 543, 546, 549
- Light sources, 301
 - accessing properties, 101
 - ambient, diffuse, and specular contributions, 241
 - area lights, 312
 - attenuation, 237–238
 - computation performed with dot products, 301
 - directional lights, 13, 236–237
 - invisible surfaces, 339
 - looping to compute contributions from all enabled, 241
 - manipulating attributes, 13
 - point lights, 13, 237–239
 - position, 152
 - reflection direction, 153–154
 - shadow maps, 340
 - shadows from, 339
 - spotlights, 239–240
 - strong, 338
 - technical illustration example, 463
 - traditional illumination model, 312
 - transforming positions, 24
 - viewing direction, 153–154
 - visible surfaces, 339
- LightColor variable, 289
- LIGHTEN blend mode, 489
- Lighting
 - artifacts, 302
 - clouds, 537–539
 - computed before texturing, 249–250
 - global parameters, 14
 - hemisphere, 311–314
 - image-based, 315–318
 - material properties and, 240–242
 - none enabled, 244
 - primary and secondary color for vertex, 13
 - RealWorldz*, 524
 - simulating in hatching example, 458–459
 - with spherical harmonics, 318–322
 - traditional equation, 462
 - two-sided, 243
 - viewing direction, 240–241
- Lighting calculations, 14, 24, 152, 190, 236
 - artifacts, 304
 - eye space, 24, 235
- Lighting designers, 323
- Lighting effects, 13, 38, 153, 316
- Lighting state, 152
- LightIntensity variable, 153, 156, 401–402, 427
- LightPos variable, 400
- LightPosition variable, 154, 289, 303
- lightVec variable, 154, 155





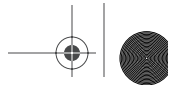
- LightWork Design, 287
- Limestone textures, 523
- #line directive, 90–91
- __LINE__ macro, 90
- Line rasterization, 479
- Linear filtering, 257
- Linear-algebraic matrix multiplication, 138–139
- Lines, uniform density, 456–458
- Linking
 - program objects, 632–635
 - shaders, 173–176
- Literal floating-point numbers, 68
- Literal integers, 68
- Local variables, defining, 159
- Local viewer computation, 241
- Locations, 196
- “Lod” functions, 142
- log function, 127
- log2 function, 127, 481
- Logical and operator (&&), 82, 86, 89
- Logical exclusive or operator (^), 82, 86
- Logical inclusive or operator (||), 82, 86, 89
- Logical not operator (!), 85, 89
- Logical operations, 17
- Logical operators, 69, 94
- Lookat point, 23
- lookup function, 343
- Lookup table operations, 481–482
- Looping, 82
- The Lord of the Rings: The Two Towers*, 544
- Low-pass filtering, 439, 678
- Lucasfilm, 418
- Luminance, 490–491
- Luminance mapping function, 481
- Luminance (LRGB) PTMs (polynomial texture maps), 376, 378
- Luxo Jr.*, 292
- L-values, 87, 677
- M**
- Macintosh environment, 2, 6
- Macros, 90
- Magi, 387
- Magnification filter, 29
- main function, 50, 65, 82, 159, 344
- Making Noises* (Peachey), 389
- Malan, Hugh, 346
- Mandelbrot, Benoit, 469
- Mandelbrot example, 467
 - color selection algorithm, 474
 - fragment shader, 472–474
 - Julia sets, 474–475
 - Mandelbrot set, 468–471
 - maximum number of iterations, 472
 - points inside and outside set, 472
 - values outside set, 472
 - vertex shader, 471–472
- Mandelbrot set, 468–471, 474
- Mapping coordinates, 26
- Marble fragment shader, 404
- Masking complex operations, 7
- mat2 data type, 51, 70, 186, 189, 192
- mat3 data type, 51, 70, 186, 189, 192
- mat4 data type, 51, 70, 186, 189, 192
- Materials
 - index of refraction, 358
 - lighting, 240–242
 - measured reflectance data, 370–372
 - non-photorealistic, 38
 - properties, 13, 240–242
 - realistic, 38
 - reflectivity, 358–359
 - specular highlights, 369
- Mathematical mappings, 481
- Matrices
 - accessing as array of column vectors, 70
 - columns as vectors, 86
 - component-wise multiplication, 138–139
 - constructors, 76
 - data types, 70
 - floating-point numbers, 70
 - indexing, 86
 - linear-algebraic multiplication, 138–139
 - manipulating, 23
 - natively supporting operations, 36
 - organized in column major order, 70
 - reading out of arguments, 77
 - selecting, 23
 - selecting columns, 51
- Matrix functions, 138–139
- Matrix multiplication operation, 153
- matrixCompMult function, 138–139
- Matte objects in technical illustration example, 463
- Matters of Light & Depth* (Lowell), 323
- max function, 129, 133–134, 155
- McMaster-Carr Supply Company, 316
- MCposition variable, 153, 156, 158, 400–401





- Member selection (.), 85
 - Meran* planet, 524
 - Meran* world, 516
 - Microsoft, 1, 544, 549
 - min function, 129, 133–134
 - Minification filter, 29
 - Mini-Mandelbrots, 471
 - Mipmap level, 29, 678
 - Mipmap textures, 28–29, 142, 398, 435, 678
 - Mipmaps
 - automatic generation, 253
 - automatically computing levels, 30
 - mix function, 130, 162, 279, 408, 435, 458, 491
 - MixRatio variable, 268
 - mod function, 128, 132, 438
 - Model space, 22, 678
 - Model transformation matrix, 23, 678
 - Modeling, 21–22, 678
 - Modeling coordinate system, 22, 678
 - Modeling coordinates, 22, 159, 326
 - Modeling matrix, 23
 - Modeling transformation, 23–24, 678
 - Modelview matrix, 152–153, 678
 - accessing current, 100
 - setting to viewing transformation, 23–24
 - transforming light positions, 234–235
 - transforming light source positions, 24
 - Modelview-projection matrix, 15, 152, 157, 678
 - MODIS (Moderate Resolution Imaging Spectroradiometer), 257
 - Modular shaders, 217–218
 - Modulus, 128
 - MojoWorld*, 393, 508–510
 - Monofractals, 508–509
 - Monsters, Inc.*, 323, 544
 - Morphing, 414–417
 - Mountainfractals, 508–509
 - Moving objects, 7
 - Multitextures, 2
 - Multifractals, 393, 508–509, 519–520, 678
 - Multi-Pass Programmable Shading* (Percy, Airey, and Ungar), 546
 - Multipass rendering position invariance, 117–118
 - Multiple render targets, 204–206
 - Multiple vertex arrays, 2
 - Multiplication operator (*), 85, 88, 138–139, 153
 - Multiplicative operators, 94
 - MULTIPLY blend mode, 489
 - Multisample buffer, 7, 679
 - Multitexturing, 30, 119
 - Multitexturing example
 - application setup, 262
 - cloud texture, 260–261
 - cloud/gloss texture, 263
 - daytime texture, 260, 263–264
 - fragment shader, 263–265
 - nighttime texture, 260, 263–264
 - reflection of sunlight off water, 260
 - vertex shader, 262–263
 - Musgrave, Ken, 389
 - Mushroom transformation, 514–515
 - MyData variable, 191–192
- N**
- Name spaces and structures, 72
 - Named attribute variables, 578–580
 - NASA Web site, 257
 - NASA/Goddard Space Flight Center, 257
 - Natural exponentiation of x , 127
 - Natural logarithm of x , 127
 - Natural phenomena, 38
 - Negative Laplacian operator, 499
 - Neighborhood averaging, 495–496, 679
 - Newell, Martin, 21
 - Nighttime texture, 260, 263–264
 - Nodes and child nodes, 227
 - Noise, 387–388, 679
 - aliasing artifacts, 393
 - animation, 417–418
 - application setup, 400
 - defining, 388–394
 - fragment shader, 401–402
 - granite, 404–405
 - higher dimensions, 395
 - marble, 404
 - number of octaves and multifractals, 520
 - OpenGL shaders, 395
 - random, 488
 - RealWorldz* values and derivatives, 517–518
 - stripe pattern, 460
 - sun surface shader, 403
 - trade-offs, 399–400
 - turbulence, 402–404
 - two-dimensional, 394–395
 - vertex shader, 400
 - wood, 405–408
 - noise function, 395, 399–400



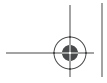


- Noise functions, 145–146, 508
 - behaving differently at different locations, 393
 - characteristics, 146
 - cubic interpolation method, 390
 - definition of, 543
 - faded to average value which aliasing artifacts occur, 393
 - folding, 402
 - fractal terrains, 510
 - ideal, 389
 - linear interpolation, 390
 - MojoWorld*, 510
 - randomness, 389
 - range of, 517
 - sampling, 512
 - texture maps, 396, 399
 - time aspect, 395
 - user-defined, 399–400
 - valid result, 398
 - varying frequency and amplitude, 390
 - Noise textures, 396–398, 510, 517
 - bilinearly filtered texture read, 521
 - creation of, 510–511
 - tiling pattern, 511–512
 - noise1 function, 146
 - noise2 function, 146
 - noise3 function, 146, 396–397, 417–418
 - noise4 function, 146
 - NoiseFunctionDerivative, 513
 - NoiseScale, 407
 - Non-mipmapped textures values, 142
 - Non-photorealistic materials, 38
 - Non-power-of-2 textures, 3
 - Nonqualified globals, 81
 - Nonqualified user-defined variables, 81
 - Nonscalar constants, 80
 - Nonwindowed system frame buffers, 6
 - NORMAL blend mode, 488
 - Normal maps, 308–309, 679
 - Normal transformation matrix, 154, 235
 - Normal variable, 270
 - Normal vectors, 11
 - Normalization, 235, 307
 - normalize function, 113, 136–137, 154, 294
 - Normalized device coordinate space, 26, 679
 - Normals
 - hatching shader, 455
 - inconsistently defined, 304
 - transformed by current normal matrix, 113
 - transforming, 154
 - transforming into eye coordinates, 152
 - Not equal operator (\neq), 89
 - not function, 89, 141
 - notEqual function, 89, 141
 - NPR (non-photorealistic rendering), 453, 679
 - NPR effects, 454
 - NumEnabledLights constant, 241
 - NVIDIA, 2, 552
 - NVIDIA developer Web site, 32, 231
- O**
- Object coordinate system, 679
 - Object coordinates, 22
 - Object space, 22, 116, 679
 - Objects
 - applying opaque image to portion of, 250
 - brick pattern, 149–164
 - default color, 273
 - deleting, 177
 - distant, 525
 - dynamic, 419
 - dynamic contour lines, 246–247
 - interreflections between, 311–312
 - mirrorlike qualities, 265
 - modeling, 21–22
 - nondeterministic shape and appearance, 419
 - origin, 22
 - polygonal representation of, 21
 - primitive particles defining volume, 419
 - random color, 273
 - three-dimensional attributes, 22
 - transforming into world coordinates, 22–23
 - unique identifier, 170
 - Occlusion factors, computing, 334–335
 - Occlusion queries, 2
 - Ocean
 - RealWorldz*, 532–537
 - reflected sky, 536
 - reflections, 532–536
 - rendering, 536–537
 - underwater caustics, 536
 - water ripples, 537
 - Octave, 390, 679
 - Offscreen buffers
 - depth buffering, 346
 - values written to, 108



- Offscreen memory, 5–6, 679
- Offset texture, 519
- ogl2brick file, 208
- ogl2demo file, 208, 257
- Old Town Square light probe image, 319
- One-dimensional texture maps, 26
- On/off animation, 412
- On/off state, 412
- OPACITY blend mode, 492–493
- Opacity variable, 190–192
- Open GL API (application programming interface), 1
- OpenGL
 - assembly language programming, 47
 - backward compatibility, 3
 - current versions, 2
 - evolution, 3–4
 - extensions, 3–4, 47
 - fixed functionality, 9, 40, 233
 - fixed-function pipeline, 2
 - frame buffer, 5–7
 - history of, 1–3
 - interaction fixed functionality, 114–120
 - maintaining state, 6
 - modifying, 3
 - normalization, 235
 - obtaining version information, 168–170
 - opening processing pipeline for user control, 2
 - operating environments, 2
 - processing commands, 4
 - processing pipeline, 37, 39
 - rasterization stage, 479
 - sending geometry data to, 9
 - setting up for rendering into multiple target buffers, 205–206
- OpenGL 1.5 to OpenGL 2.0 GLSL Migration Guide, 660–663
- OpenGL 2.0 Shading Language White Paper, Version 1.0* (Baldwin), 165, 233
- OpenGL API execution model, 4–5
- OpenGL ARB (Architecture Review Board), 1–2, 32
- OpenGL driver, 54–55
- OpenGL extension registry Web site, 3
- The OpenGL Graphics System: A Specification (Version 2.0)* (Segal and Akeley), 32, 33, 63, 96, 121, 148, 165, 213
- OpenGL Performer, 229
- OpenGL Performer Web site, 231
- OpenGL Programming Guide, Fifth Edition: The Official Guide to Learning OpenGL* (Shreiner, Neider, Davis, and Woo), 32, 213
- OpenGL Reference Manual, Fourth Edition: The Official Reference to OpenGL*, 32, 213
- OpenGL Shader Language, 679
- OpenGL Shader Language API, 679
- OpenGL shaders, 36, 544, 547–549, 679
 - code to transfer CIE values to RGB, 483
 - code to transform RGB values to CIE, 483
 - cube map, 265
 - noise, 395
- OpenGL Shading Language, 544
 - added features, 50–52
 - additions from C++, 52
 - basics, 35–37
 - built-in
 - functions, 48
 - math functions, 547
 - variables, 546
 - C programming language, 50, 53
 - compiler front end, 225–226
 - compiler/linker, 54–57
 - cross-vendor assembly language, 60
 - data types, 546
 - defining capabilities for hardware, 549
 - as descendent of RenderMan, 547
 - design considerations, 47–50
 - ease of use, 49
 - exposing hardware capabilities, 48
 - exposing performance of graphics hardware, 49
 - exposing programmability of hardware, 549
 - fragment shaders, 546
 - hardware independence, 48
 - as high-level language, 60–61
 - key benefits, 59–61
 - keywords, 547
 - libraries or executables, 61
 - longevity, 49
 - mapping onto graphics hardware, 546
 - modular programming support, 61
 - obtaining version information, 168–170
 - open, cross-platform standard, 60
 - OpenGL environment, 48
 - overview, 47–54
 - parallel processing, 49
 - precedence of operators, 547



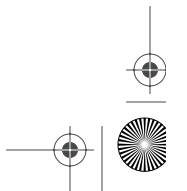


- processing numerical data, 53
- RenderMonkey, 223
- runtime compilation, 59
- standardization, 49
- syntax, 65
- tight integration with OpenGL, 59
- transformations, 234–235
- type qualifiers, 36
- types, 36
- unnecessary language features, 49
- variable nodes, 224
- vector types, 50–51
- version 110, 92
- version number, 90
- vertex shaders, 546
- The OpenGL Shading Language, Version 1.10*
(Kessenich, Baldwin, and Rost), 63, 95, 96,
121, 147, 165, 213
- OpenGL Shading Language API, 57–59, 167
- implementation-dependent values, 207
- OpenGL state
 - accessing current, 42, 109–112
 - automatic tracking mechanism, 101
 - client-side, 8
 - controlling with scene graphs, 227–228
 - derived, 112
 - fragment shaders, 97
 - maintaining, 6
 - manipulating, 108
 - modifying, 196
 - on or off, 8
 - pushing and popping values on stack, 8
 - server-side, 8
 - settings, 8
 - texture units, 27
 - tracking, 196
 - user-defined, 67
 - vertex shaders, 97
- OpenGL SuperBible, Third Edition* (Wright and
Lipchak), 32–33
- OpenGL utility library, 24
- OpenGL.org Web site, 32
- OpenSceneGraph Web site, 231
- OpenSG, 229
- OpenSG Web site, 231
- Operating environments, 2
- Operations
 - accelerating, 123
 - component-wise, 87–90
 - indexing, 86
 - operators, 85–86
 - swizzling, 87
- Operators, 85–86
 - applied to vector, 87–88
 - vector types, 50
- Optimizations, 91
- Optimize pragma, 91
- Optional set of imaging functionality, 2
- Origin, 22
- Orlano, Marc, 544
- Orthonormal basis, 303
- OSG (OpenSceneGraph), 229
- out qualifier, 54, 83
- Overhanging terrain, 513–517
- OVERLAY blend mode, 490–491
- Overloading
 - built-in functions, 52
 - functions, 36, 52, 82, 123
- Overriding
 - built-in functions, 84–85
 - earlier directives, 92
- P**
- Parallel projection, 25
- Parameters, 83, 440
- Parsing, 225, 679
- Particle systems, 418–426, 680
 - application setup, 420–423
 - assumptions, 419
 - complexity, 419
 - confetti cannon vertex shader, 423–424
- Particle Systems—A Technique for Modeling a Class
of Fuzzy Objects*, 419
- Particles, 419–425
- ParticleTime variable, 423
- Pass nodes, 224
- PCs, 2
- Peachey, Darwyn, 149, 272, 444
- Pearl Harbor*, 338
- Pellacini, Fabio, 323, 343
- Percentage variable, 274
- Per-fragment operations, 16–17, 680
- Perlin, Ken, 145, 387–388, 543
- Perlin heterofractal, 509
- Perlin noise, 393, 460, 508, 510–512, 680
- Perspective divide, 26, 40
- Perspective projections, 15, 25
- Perspective view, 15





- Perspective-correct projective texture
 - coordinates, 341–342
- Perturbation normal, 307
- Perturbation vector, 307
- Per-vertex arbitrary data, 10
- Per-vertex attributes, 185
- per-vertex attributes, 10
- Per-vertex lighting, 154
- Per-vertex operations, 12–14
- Per-vertex values, interpolating, 67
- Phong exponent, 317
- Phong lighting equation, 524
- Phong lighting model, 464
- Photorealism, 453, 680
- Pixar Animation Studios, 322, 543
- Pixel group, 19, 680
- Pixel ownership test, 17, 680
- Pixel packing, 21, 680
- Pixel Pipeline shader, 233
- Pixel rectangles, 680
 - color, 18
 - grayscale, 18
 - pixel transfer, 19–20
 - rasterization, 479
 - read control, 20–21
- Pixel shaders and HLSL (High-Level Shader Language), 551
- Pixel transfer, 19–20, 680
- Pixel unpacking, 19, 21, 680
- Pixel zoom, 480
- PixelFlow, 543–544
- Pixels, 5, 433–434
 - visible or obscured, 17
 - window coordinate system, 26
- Planets
 - fractal terrains, 508–510
 - overhanging terrain, 513–517
 - shading, 507–508
 - surface normals, 513
 - terrain-rendering structure, 506–507
- Point light sources, 13
- Point lights, 237–239
 - attenuation, 237–238
 - computation, 238–239
 - direction of maximum highlights, 237
 - spotlight cutoff angle, 241
- Point primitive, 101
- Point R, 532
- Point rasterization, 2, 479
- Point sampling, 434, 680
- Point size, clipping, 115
- Point size mode, 115–116
- Point sprites, 3, 16, 425, 681
- Pointers, 53, 627–628
- PointLight function, 243
- Polygon mode, 40
- Polygon offset, 40
- Polygon primitives, culling, 15
- Polygon rasterization, 479
- Polygons, 16, 21
- Polynomial texture mapping with BRDF data, 375–384
- Position invariance, 117–118
- Position values, accessing, 50
- position variable, 159, 471
- Position vector, 69
- position.x variable, 445
- Positive square root, 127
- Postfix decrement operator (--), 85
- Postfix increment operator (++), 85
- pow function, 127, 481
- #pragma directive, 90–91
- Pragmas, reserving for future use, 91
- Prefix decrement operator (--), 85
- Prefix increment operator (++), 85
- Preprocessor, 90–94
- Primary color, 242
- Primitives, 681
 - antialiasing, 16
 - assembly, 14, 40, 66, 681
 - clipping, 14–15
 - drawing large number with single function call, 11
 - processing, 14–15
 - specifying type, 14
 - starting and ending, 10
- printShaderInfoLog function, 209
- Procedural texture shaders, 285–287, 681
- Procedural textures, 38
- Procedural texturing, 285, 681
- Processing pipeline, 8–9
- Program objects, 55–56, 681
 - active attribute variable information, 598–600
 - active uniform variable information, 601–604
 - attaching shader objects, 55, 173, 209–210, 576–577





- creation of, 583–584
 - defining, 173
 - deleting, 177, 587–588
 - detaching shader objects, 177, 591–592
 - failure of linking, 175
 - handles of shader objects attached, 605–606
 - information log, 175, 182, 210, 612–613
 - installing as part of rendering state, 645–648
 - linking, 174, 632–635
 - listing active uniform variables, 201–203
 - loading uniform variables into, 197–198
 - names, 184, 629–630
 - parameters, 609–611
 - queriable parameters, 179–180
 - querying list of shader objects attached to, 183
 - querying state, 179–180
 - state, 197
 - validating, 649–650
 - value of uniform variable for, 638–644
 - Programmable fragment processors, 39
 - Programmable processors
 - fragment processors, 38–39, 43–47
 - stream processing, 39
 - successfully installing executable, 176
 - vertex processors, 38–39, 40–43
 - Programmable shaders, 233
 - Programmable vertex processors, 39
 - Programs, 36, 681
 - diagnostic information, 206–207
 - ill-formed, 94–95
 - installing as part of rendering state, 176
 - managing shader objects, 176
 - Projection matrix, 25, 681
 - Projection transformation, 25, 681
 - Projective texturing, 142
 - PTMs (polynomial texture maps), 375–378, 681
 - Pulse train, 444, 681
- Q**
- Qualified variables, 67
 - Qualifiers, 51, 78
 - absent qualifier, 81
 - attribute qualifiers, 79
 - constant qualifiers, 80–81
 - type, 78
 - uniform qualifiers, 79
 - varying qualifiers, 79–80
 - Queriable values, 575
 - Query functions, 178–184
 - Querying
 - attribute binding for vertex shader attribute variable, 189–190
 - locations of user-defined uniform variables, 196–197
 - shader objects, 183
 - user-defined variable current value, 199–200
- R**
- rad variable, 427–428
 - Radians, 125
 - radians function, 125
 - Ramamoorthi, Ravi, 318–319
 - RandomRotate variable, 274
 - RandomScale variable, 274
 - Raster position, 20, 117, 682
 - Rasterization, 15–16, 20, 682
 - Rasterization stage, 479
 - Ratio between reflected light and refracted light, 359
 - Read control, 20–21, 682
 - Realism, 453
 - Real-time atmospherics, 525–529
 - Real-Time Stroke-based Halftoning* (Freudenberg), 454
 - RealWorldz*, 505
 - 2D instead of 3D noise, 510
 - Altgrad map, 522–524
 - atmospheric effects, 525–532
 - caustic effects, 536
 - clouds, 537–539
 - DLU node, 509
 - features, 505–506
 - fractal terrains, 508–510
 - function tree, 509, 519–521
 - implementation, 517–525
 - internals, 506–517
 - lighting, 524
 - LOD (level-of-detail), 507
 - Math node, 509
 - Multifractal node, 509
 - mushroom transformation, 514–515
 - node types, 509
 - noise texture creation, 510–511
 - noise values and derivatives, 517–518
 - ocean, 532–537
 - overhanging terrain, 513–517
 - performance considerations, 524–525



*RealWorldz, continued*

planet creation, 506–517
 planetary landscapes, 506
 postprocessing height field terrain, 514
 quadtree structure, 506–507
 separate terrain function trees, 516–517
 shading, 507–508
 sky color, 529–532
 smoke, 539
 surface normals, 513
 terrain color, 521–522
 texture map, 507, 525
 textures, 505–506
 tile set noise, 511–512
 tile sets, 518–519
 VBO (vertex buffer object), 506

Reciprocal of positive square root, 127

Recursive function, 469

Reeves, Bill, 418–419

reflect function, 137, 154, 267, 360, 373

Reflectance model, 368

ReflectDir variable, 268

Reflected vector, 360

Reflection vector, 154, 156

reflectionDir vector, 271

Reflections

- AlienRockArt* planet, 532
- computing, 248, 368–375
- direction, 137, 153–154, 271
- divider cone, 534
- horizon cone, 534
- ocean, 532–536
- point R, 532
- properties, 534
- reducing, 338
- terrain, 532–536

Reflections on Spheres and Cylinders of Revolution (Glaeser), 532

Reflectivity, 358–359

reflectVec variable, 156

refract function, 137, 358, 360

Refracted vector, 358, 360

Refraction, 358–363

- Cg shaders, 359
- ratio of indices, 137
- wavelength dependent, 362

Refraction effect, 358

Regular patterns, 287–288

- amount of fuzz (blurriness), 291

- antialiased transitions between stripe colors, 291
- smooth transitions between stripe color and background color, 290
- stripes fragment shader, 290–291
- stripes vertex shader, 289–290

Relational and equality operators (>, <=, >, >=, ==, !=), 139

Relational operations based on vectors, 52

Relational operators (<, >, <=, and >=), 69, 85, 89, 94

Render targets, multiple, 3, 204–206

Rendering, 5, 479, 682

- advanced effects, 38
- double buffering, 6–7
- installing program as part of state, 176
- installing program objects as part of state, 645–648
- objects with mirrorlike qualities, 265
- ocean, 536–537
- passes, 223
- sampling pixels multiple times, 7
- scene graphs, 227
- windows supporting, 6

Rendering algorithms

- understanding, 216
- user-created, 2

Rendering Outdoor Light Scattering in Real Time (Hoffman and Preetham), 526

Rendering pipeline, 8, 682

RenderMan, 340, 545–547

The RenderMan Companion: A Programmer's Guide to Realistic Computer Graphics (Upstill), 147–148

RenderMan Interface Specification, 148, 543, 544–547

RenderMan shader, 149

RenderMan Shading Language, 546

RenderMonkey, 222–225

Repeat wrapping behavior, 257

Repeating patterns, 518–519, 524

require behavior, 93

Rescaling normalization, 235

Resolution, increasing and antialiasing, 436–437

Restricted to cone of light, 239

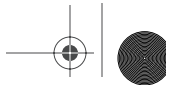
return keyword, 83

RGB color space, 481

RGB PTMs (polynomial texture maps), 376, 378

RGB representation of color, 365





- RGBA (red, green, blue and alpha component), 19
- RGBA texture, 396
- Rideout, Philip, 257, 329, 338, 341, 343, 415
- Ring* planet, 524
- Ring world, 509
- Ripple texture map, 537
- Ritter, Brad, 375
- Rivero, Michael, 467
- RO1 variable, 278
- Runtime compilation, 59
- R-values, 87, 682

- S**
- sample3D data type, 203
- sampler data type, 141
- sampler1D data type, 51, 71, 141, 203, 255
- sampler1DShadow data type, 71, 203
- sampler1DShadow sampler, 119
- sampler2D data type, 51, 71, 141, 203, 255
- sampler2DShadow data type, 71, 203
- sampler2DShadow sampler, 119
- sampler3D data type, 71, 255
- samplerCube data type, 71, 203, 255
- Samplers, 51, 71–72, 203–204, 255, 682
- samplerShadow1D data type, 255
- samplerShadow2D data type, 255
- SamplesPerCell variable, 275
- Saturation, 485
- Sawtooth wave, 438
- Scalar Booleans, 82, 89
- Scalar components, 86
- Scalar constants, 80
- Scalars
 - binary operation, 88
 - data types, 67–69
- Scale-and-bias, 481
- ScaleFactor variable, 272
- Scaling modeling coordinates, 164
- Scanning, 682
- Scene graphs, 227–229, 682
- Scenes
 - global ambient lighting value, 14
- Schlick, Christophe, 359
- Scissor test, 17, 682
- Scope of variables, 74
- SCREEN blend mode, 490
- Secondary color, 242
- Segal, Mark, 4
- Selection, 82
- Selection of flat or smooth shading, 40
- Selection operator (?), 82, 86
- Self-shadows, 344
- Semantic analysis, 225, 682
- Sequence operator (,), 86, 90
- Server-side stack, 8
- Server-side state, 8
- setNoiseFrequency function, 396
- SGI (Silicon Graphics, Inc.), 1–2, 544
- SGI OpenGL Web site, 33
- SGI prefix for extensions, 3
- Shader development
 - analyzing algorithm, 219
 - built-in function, 219
 - general principles, 215–218
 - modularity, 217–218
 - person writing, 222–223
 - progressively adding complexity, 216–217
 - reducing complexity, 223
 - reviewing information logs, 220
 - set of parameters, 217
 - simplicity, 217
 - testing, 217
 - textures to encode complex functions, 220
 - understanding problem, 216
 - vectors, 220
- Shader development tools
 - OpenGL Shading Language compiler front end, 225–226
 - RenderMonkey, 222–225
- Shader languages
 - Cg, 544, 552–554
 - HLSL, 544
 - OpenGL Shader, 544
 - OpenGL Shading Language, 544
 - Stanford Real-Time Shading Language, 544
- Shader objects, 54, 682
 - attaching to program objects, 55, 173–174, 209–210, 576–577
 - compiling, 172–173, 581–582
 - container for, 55
 - creation of, 170–172, 585–586
 - deleting, 177–178, 589–590
 - detaching from program objects, 177, 591–592
 - empty, 171, 209



**Shader objects, *continued***

- GL_COMPILE_STATUS parameter, 209
- handles of attached for program objects, 605–606
- information about, 178
- information log, 172–173, 181, 616–617
- linking, 173–176
- names, 184, 631
- obtaining current shader string from, 180–181
- parameters, 614–615
- queriable parameters, 178–179
- querying, 183
- replacing source code, 636–637
- source code string, 618–619
- support for multiple, 55–56

ShaderGen, 233
Shaders, 36, 134, 682

- accessing current, 109
- accessing state, 51
- accessing texture maps, 254–256
- algorithm defining, 285
- analyzing code, 219
- animation effect, 411–412
- bump-mapping, 216
- classification of, 543
- common interfaces, 61
- compiling and annotating, 91
- compiling to machine code, 55
- computational frequency, 218
- computations done once on CPU, 219
- debugging, 91
- declaring OpenGL Shading Language version, 91–92
- defining source code, 171
- description of effect, 150
- developing, 91
- diagnostic messages returned from compiling, 56
- easier to understand and maintain, 123
- example pair, 65–67
- failing compilation, 209
- first pass of volume shadow algorithm, 349–350
- getting information into and out of, 78
- highly customized, 222
- implementing as collection, 61
- improving performance, 219
- inability to manipulate sampler values, 72

- information log, 209
- interface, 78–81
- length, 36
- linking, 173–176
- lookup table operations, 481
- managing input and output, 51
- modifying attributes, 79
- modifying behavior, 196
- noise, 395
- overriding built-in function, 84–85
- parameterized with uniform variables, 42
- passing to OpenGL as strings, 208
- passing to renderer through RenderMan interface, 546
- performance considerations, 218–220
- programmable, 233
- qualifiers, 78–81
- range of effects possible with, 37–38
- reading and writing variables, 81
- reasons for using, 37–38
- receiving samplers, 71
- referring to existing state, 48
- rendering procedural brick pattern, 149
- runtime checking, 206–207
- saving instruction space, 219
- second pass volume shadow algorithm, 350–354
- sharing same array, 74
- simpler to develop, 123
- source code, 54
- source strings, 56
- two-dimensional images, 479
- uniform variables, 100–101
- unpacking noise texture sample into function value, 517–518
- usage, 173–176

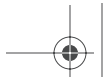
Shading effects, 106
Shading languages

- chronology of, 543–544
- compiler, 55
- HLSL (High-Level Shader Language), 549–552
- OpenGL Shader, 547–549

Shading planets, 507–508
Shadow Map Antialiasing, 343
Shadow mapping, 683

- aliasing artifacts, 339
- depth comparison, 339
- lights capable of casting shadows, 346





- matrix transforming vertex positions for
 - light source, 340–341
 - modeling matrix, 341
 - projection matrix, 341
 - rendering passes for light sources, 346
 - scale and bias matrix, 341
 - view matrix, 341
- Shadow mapping algorithm, 339
- Shadow maps, 338–340, 683
 - application setup, 340–341
 - bias depth values, 339
 - depth precision problems with illuminated faces, 340
 - depth texture, 343
 - fragment shader, 343–346
 - lookup, 343
 - per-light passes, 339
 - precision problems on surfaces facing away from light, 340
 - sampling multiple times, 344
 - vertex shader, 341–342
- shadow texture, 142
- Shadow volume, 347–348
- shadow1D function, 119, 144, 255–256
- shadow1DLod function, 144
- shadow1DProj function, 144
- shadow1DProjLod function, 144
- shadow2D function, 119, 144, 256, 344
- shadow2DLod function, 144
- shadow2DProj function, 144, 343
- shadow2DProjLod function, 144
- Shadowing algorithm, 339
- Shadows, 253, 333
 - aliasing artifacts, 343
 - artifact-free, 340
 - comparison functions, 2
 - complex, 334
 - compositing, 347
 - from light sources, 339
 - object-generated, 344
 - soft, 334
 - soft with deferred shading, 346–347
 - storing depth values, 2
 - uses for, 333
 - well-defined, 338
- Sharpening images, 499–500
- Sharpness, 486
- Shirley, Peter, 463
- sign function, 128, 130
- sin function, 124–125, 426
- Sine, 125
- sine function, 426, 509
- Sine functions, 84–85
- Sine noise, 510
- Single-buffered windows, 7
- sizeof operator, 94
- Sky
 - color texture map lookup in fragment shader, 530–531
 - optical depth, 530
 - reflected in ocean, 536
 - shading, 529–532
- Sliders, 158
- Smoke, 539
- Smooth shading, 15, 683
- Smoothing
 - blending modes, 495–497
 - general convolution shader, 496–497
 - images, 495–497
 - neighborhood averaging, 495–496
- Smoothing filters, 683
- smoothstep function, 130, 135, 164, 291, 296–297, 324, 413, 435, 439–440, 442, 459
- Snell's law, 358
- Snow* planet, 522, 529
- Soft fill light, 338
- SOFT LIGHT blend mode, 491
- Source code, 170–173
- Source image interpolation, 484
- Source strings, 90–91
- Special input variables, 104, 106
- Special operations, 69
- Special output variables, 101–102, 107–108
- Specular color component, 14
- Specular component, 155, 368
- Specular contribution, computing separately, 242
- Specular glint, 366
- Specular highlights, 14, 156, 369
 - applied after texturing, 242
 - Gooch shading, 464
- Specular lighting effects, 218
- Specular reflection, 152–154, 156, 260, 368
- Specular variable, 289
- SpecularColor variable, 289
- specularContribution constant, 156
- SpecularFactor variable, 303, 307
- sphere, 416





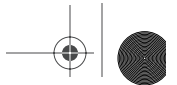
- Sphere map computations, 248
 - Sphere mapping, 265, 683
 - Sphere morph vertex shader, 415–417
 - Spherical harmonics lighting, 318–322, 337
 - Spline function, 414
 - SpotLight function, 243
 - Spotlights, 13, 239–240
 - sqrt function, 127, 352
 - Stöckli, Reto, 257
 - Stam, Jos, 363
 - Standard attributes, 99–100, 190
 - Stanford Real-Time Shading Language, 544
 - Stanford Shading Language, 55
 - Star Trek II: The Wrath of Khan*, 418
 - Star Wars Episode 1: The Phantom Menace*, 544
 - StartRad variable, 427
 - StartTime variable, 423
 - State of the Art in Hardware Shading*, 304
 - static variables, 81
 - STDGL pragma, 91
 - STDGL token, 91
 - Steep slopes and textures, 513–514
 - Stencil buffer, 7, 17, 19, 683
 - Stencil components, initializing, 18
 - Stencil functionality, 3
 - Stencil test, 17, 683
 - Stencil wrapping behavior, 2
 - step function, 130, 134, 161, 164, 439, 456, 458
 - Stereo viewing, 7
 - Stored texture shader, 287
 - Stored textures, 287
 - Stream processing, 39
 - Strings, 53
 - Strings of characters, concatenating, 56
 - Stripe patterns
 - colors smoothly “fuzzed” boundary in transition region, 291
 - noise, 460
 - Stripe shader, 287–288
 - Stripes fragment shader, 290–291
 - Stripes vertex shader, 289–290
 - Structure field operator (\cdot), 197
 - Structure variables, 80
 - Structures, 72
 - bit fields, 53
 - constructors, 76
 - grouping collection of parameters, 112
 - SUBTRACT blend mode, 492
 - Subtraction operator ($-$), 85, 88
 - Sun Microsystems, 2
 - Sun surface fragment shader, 403
 - Superellipse function, 323–324
 - Superellipses, 324, 326
 - superEllipseShape, 326
 - Supersampling, 683
 - Surface normals, 41, 154–155, 513, 539
 - eye space, 24–25
 - transforming, 235
 - Surface shaders, 543, 546, 549
 - SurfaceColor variable, 303, 307
 - Surface-local
 - coordinates, 303
 - eye direction, 306
 - light direction, 306
 - transformation matrix, 306
 - Surface-local coordinate space, 302, 683
 - Surfaces
 - computing color, 242
 - computing reflection, 368–375
 - reconstructing color under varying lighting, 375
 - rendering with evaluators, 12
 - switch keyword, 82
 - Swizzle, 684
 - Swizzle operator (\cdot), 85, 87
 - Swizzling, 87
 - Symbolic constants, 8
 - Syntactic analysis, 225, 684
 - System
 - driver model, 54–56
 - overview, 54–59
- T**
- tan function, 125
 - Tangent space, 303, 684
 - tangent variable, 372
 - Tangent vectors, 302–304
 - Tangents, 125, 304
 - Target buffers, rendering into multiple, 205–206
 - Target image extrapolation, 484
 - Taylor, Philip, 312
 - Technical illustration example, 462
 - application setup, 466
 - common characteristics for color, 463
 - effects, 463
 - fragment shader, 466–467





- light source, 463
- matte objects, 463
- vertex shader, 466
- warmth or coolness of color, 463
- Technical illustrations, 462
- Tejada, Antonio, 426
- Temperature values, 66
- Temporal aliasing, 435, 684
- Ternary selection operator (?:), 89–90
- Terrain
 - color, 521–522
 - high frequency features, 515–516
 - reflections, 532–536
 - separate terrain function trees, 516–517
 - steep slopes and textures, 513–514
- Terrain function tree, 519–521
- TexCoord variable, 381, 383
- Texels, 29, 684
- texName texture object, 257
- texNT function, 521
- Text application, 684
- Texture access, 31, 46, 684
- Texture access functions, 141–144
- Texture application, 31
- Texture bombing, 676, 684
- Texture coordinate sets, 118–119
- Texture coordinates, 41, 255, 307
 - automatically generating, 30–31
 - built-in array, 73
 - clamping to range, 279
 - generating, 246–248
 - perturbing, 426
 - random scaling, 279
 - reference frame from, 272
 - simple environment mapping, 247
 - transforming, 235
- Texture environment
 - color, 250
 - parameters, 30
- Texture environment function, 30
- Texture formats, 2
- Texture functions
 - built-in, 249, 482
 - shadow variants, 142
- Texture generation functions, 31
- Texture image units, 113, 118–120
- Texture lookups, 71–72, 118
- Texture mapping, 16, 26, 684
 - example, 256–259
 - extending scope, 253
 - limits related, 119
- Texture maps, 376
 - accessing from shaders, 51, 254–256
 - cube maps, 26
 - daylight colors, 260
 - defining behavior of filtering operation, 42
 - drawing shadow, 348
 - fragment shader, 104
 - implementing noise function, 399
 - index values, 271
 - levels of gray, 272–273
 - locations accessed, 255
 - no lighting, 260
 - noise function, 396
 - one-dimensional, 26
 - parameters, 46
 - RealWorldz*, 507
 - storing previously computed noise function, 395
 - three-dimensional, 26
 - time to generate, 525
 - two-dimensional, 26
 - vertex shader, 99
- Texture matrix stack, modifying, 31
- Texture memory, 16, 249, 684
 - accessing arbitrary number of times, 46
 - directly updating, 20
 - images for, 18
 - new uses for, 38
 - vertex processors reading from, 42
- Texture objects, 2, 28–30, 203, 253, 684
- Texture pattern, 272
- Texture target, 28
- Texture units, 27, 30, 258, 685
- Texture values, 51
- Texture wrap mode, 253
- texture1D function, 87, 143, 255
- texture1DLod function, 143
- texture1DProj function, 143
- texture1DProjLod function, 143
- texture2D function, 143, 255
- texture2DLod function, 143
- texture2DProj function, 143
- texture2DProjLod function, 143
- texture3D function, 143, 255
- texture3DLod function, 143





- texture3DProj function, 143
- texture3DProjLod function, 143
- Texture-access functions, 255
- Texture-combine environment mode, 250–251
- textureCube function, 144, 255–256
- textureCubeLod function, 144, 256
- Texture-enables and fixed function hierarchy, 119
- Textures, 16
 - accessing from shader, 46
 - accessing multiple simultaneously, 253
 - aliasing, 435
 - application, 249–251
 - color components, 309
 - comparison operation, 29
 - compressed image formats, 28
 - creation of, 28
 - defined algorithmically, 286
 - encoding complex functions, 220
 - fixed to geometric model, 246
 - generated procedurally, 285
 - level-of-detail, 142
 - modifying, 28
 - no fixed area or resolution, 286
 - non-power-of-2, 2–3
 - normal perturbation values, 308
 - priority to be assigned to, 29
 - projecting on object, 142
 - properties, 142
 - RealWorldz*, 505–506
 - RGB values, 250
 - setting state, 141
 - setting up for use, 254
 - sizes not restricted to powers of 2, 28
 - steep slopes, 513–514
 - switching between, 28
 - three-dimensional, 253
 - transformation, 31
 - type accessed, 203, 255
 - uses for, 253–254
- Texturing, 26–31
 - additional capabilities, 2
 - lighting computed before, 249–250
 - undefined results, 142
- Texturing and Modeling: A Procedural Approach, Third Edition* (Ebert et al.), 149, 165, 213, 230–231, 272, 389, 393, 444
- Texturing example, 256–259
- Theoretical BRDF models, 370
- Three-dimensional attributes, 22
- Three-dimensional texture maps, 26
- Threshold animation, 413
- Tile set noise, 511–512
- Tile sets, 518–519
- Tileable 3D texture and Perlin noise, 460
- Time variable, 423, 455
- T&L (transformation and lighting), 13, 684, 685
- tnorm variable, 154–155
- Tokens, 91
- Tom Nuyden’s Web site, 32
- Topmost matrix, 24
- Toy ball, 292
 - application setup, 293
 - defining sphere, 293
 - diffuse-only lighting, 298
 - dot product, 296
 - fragment shader, 294–299
 - fragments in stripe pattern, 293
 - half-space computations, 295–296
 - inorout counter, 297
 - lighting calculation, 297–298
 - smoothly antialiased transition, 296
 - specular highlight, 298
 - star pattern, 292–293
 - surface color for fragment, 297
 - surface location, 294–295
 - surface points, 297
 - uniform variables, 293
 - vertex positions, 293
 - vertex shader, 294
- Toy Story*, 323, 544
- Traditional illumination model, 311–312
- Traditional lighting equation, 462
- Traditional lighting model, 463
- Transformation matrix, 302
- Transformations, 21, 234–235
- Transformed vertices, 67
- Transforming, 235
- Trigonometric operations, 52
- Trigonometry functions, 124–126
- Tron*, 387
- true literal Boolean constant, 68
- Tufte, Edward, 462
- Turbulence, 402–404, 510, 685
- Two-dimensional arrays, 494
- Two-dimensional images, 479
- Two-dimensional texture maps, 26





- Two-sided color mode, 114–115
- Two-sided lighting, 40, 243
- Type casting without conversion, 53
- Type conversions, 77
- Type matching, 75
- Type qualifiers, 36, 40
- typedef keyword, 72
- Types, 36
 - larger and smaller, 53
 - strict with, 36
- U**
- überLight controls, 323
- überlight model
 - fragment shader, 326–329
 - positioning lights, 323
 - vertex shader, 325–326
- überlight shader, 322–324, 329
- Unary negation (-) operator, 85, 88
- Unary operators, 94
- #undef directive, 90
- Under the Shade of the Rendering Tree* (Lander), 464
- Underwater caustics, 536
- Uniform line sensity, 456–458
- uniform qualified sampler, 71
- Uniform qualified variables, 67, 78–79
- Uniform qualifiers, 78–79, 547
- Uniform variables, 78–81, 100, 106, 152, 158, 195, 685
 - accessed when shader is executed, 201
 - application-defined, 101
 - built-in, 42, 46, 51, 97, 105, 108–113, 196, 202
 - changes to, 42
 - declared as arrays, 199
 - declared as structures, 199
 - declaring, 195–196
 - defined as sampler types, 198
 - fragment shader, 104
 - initializing, 75, 210
 - list of active, 201–203
 - loading defined as sampler types, 203
 - loading into program object, 197–198
 - location of, 622–623
 - return value, 620–621
 - sampler type, 255
 - setting parts, 200–201
 - setting values, 208
 - size, 202
 - user-defined, 42, 46, 97, 105, 202, 233
 - value for program object, 638–644
 - vertex shader, 99
- union enum keyword, 72
- Unions, 53
- Unit vectors, 154
- Units of measurement, 22
- UNIX-based systems, 2
- Unqualified variables, 81
- Unsharp masking, 486, 499–500, 685
- User clip planes, 116, 249
- User clipping, 25, 40, 102, 116–117, 249, 685
- User programmable antialiasing methods, 38
- User-defined attribute variables, 41–42, 97, 168
 - data types, 192
 - mapping generic vertex attributes to, 190–192
- User-defined attributes, 79
- User-defined clipping planes, 15
- User-defined functions, 399
- User-defined noise functions, 399–400
- User-defined state, 67
- User-defined uniform variables, 42, 46, 97, 105, 202, 233
 - querying current value, 199–200
 - querying locations, 196–197
 - updating value, 196
- User-defined varying variables, 43, 45, 103
- Utah teapot, 21–22
- V**
- Value noise, 389, 685
- Values, 77, 133–134
- Vanishing point, 25
- Variable nodes, 224
- Variables
 - attribute, 81
 - built-in, 51, 78, 81, 97, 101, 152, 546
 - declaring, 36, 52, 74–75
 - global scope, 78, 102
 - initializing, 53, 75
 - naming, 74
 - qualified as const, 80–81
 - reading and writing, 102
 - scope, 74
 - uniform, 81
 - unqualified, 81





- varying mechanism, 81
- Varying qualified variables, 67, 78–80
- varying qualifiers, 78–80, 547
- Varying variables, 43, 51, 78, 104, 685
 - built-in, 43, 102–103, 105
 - fragment shader, 104, 163
 - initializing, 75
 - interpolated values, 80
 - user-defined, 43, 45, 103
 - vertex processor, 99
 - vertex shader, 163
- Varyings, 79–80
- VBO (vertex buffer object), 506
- vec2 data type, 50, 52, 69, 101, 150, 192
- vec3 data type, 50, 52–54, 69, 101, 153, 192
- vec4 data type, 50, 52, 69, 153, 192, 272
- Vector components, 86
- Vector relational functions, 139–141
- Vectors
 - accessing components, 50
 - binary operation, 88
 - built-in constructors, 76
 - built-in functions, 69
 - built-in variables, 69
 - compile-time checking, 70
 - component access, 69
 - component selection names, 70
 - data types, 69–70
 - Euclidean distance between two points, 137
 - indexed as zero-based arrays, 70
 - indexing, 86
 - length, 136
 - with length of 1, 136
 - names available for selecting components, 69
 - natively supporting operations, 36
 - normalizing, 154
 - operators, 50, 87–88
 - shader development, 220
 - special operations, 69
 - swizzling components, 87
- Velocity variable, 423
- #version directive, 91–92
- __VERSION__ macro, 90
- Vertex arrays, 2, 11
 - drawing geometry, 99
 - interface, 185
 - specifying different texture coordinate arrays, 30
 - storing data in server-side memory, 12
- Vertex attribute array
 - defining, 657–659
 - enabling or disabling, 596–597
- Vertex attributes, 10–11, 41–42, 185–187, 651–656, 686
- Vertex buffer objects, 2, 185
- Vertex noise, 417–418
- Vertex positions, 41
 - built-in attributes, 421
 - current array, 11
 - transforming to eye coordinates, 360
- Vertex processing, 2, 12–14, 35–36, 686
- Vertex processors, 38, 41, 686
 - built-in varying variables, 102–103
 - data values as inputs, 40
 - data values produced by, 40
 - executing vertex shaders, 98
 - fixed functionality, 39
 - operating on one vertex at a time, 42
 - output, 43
 - passing data values from application, 42
 - programmable, 39
 - reading from texture memory, 42
 - sending standard OpenGL vertex attributes, 41
 - special output variables, 101–102
 - uniform variables, 100–101
 - user-defined varying variables, 103
 - varying variables, 99
 - vertex attributes, 99–100
 - vertex shaders, 40
- Vertex shaders, 36, 40, 686
 - attribute variables, 40–41, 99
 - attributes, 97
 - bias parameter, 142
 - BRDF PTMs, 379–381
 - brick pattern, 151–157
 - built-in
 - attribute variables, 163
 - constants, 99, 113
 - variables to access standard types of data, 184
 - varying variables, 97
 - bump mapping, 306
 - characteristics that don't change, 218
 - chromatic aberration effect, 362–363
 - common functions, 126, 128–134
 - communicating results to fragment shader, 79–80





- communicating to subsequent processing, 97
- computing homogeneous vertex position, 153
- computing vertex position in eye
 - coordinates, 153
- confetti cannon, 423–424
- cube mapping example, 266–267
- current state, 42
- defining, 151–152
- diffraction effect, 367–368
- displacement mapping algorithms, 42
- distance attenuation, 116
- empty shader objects, 209
- environment mapping example, 269
- example, 65–67
- executing, 42, 98–99
- explicitly binding vertex attribute to
 - attribute variable, 188–189
- exponential functions, 126–127
- fog factor, 246
- fragment shader, 65–67
- Fresnel reflection/refraction effect, 361
- generating shadows, 342
- generic attributes, 100
- generic vertex attributes, 42
- geometric functions, 136–138
- glyph bombing, 276–277
- hatching example, 455–456
- hemisphere lighting, 314–315
- hemisphere lighting with ambient occlusion, 335–336
- HLSL (High-Level Shader Language), 551
- identity mapping of input geometry, 494
- image-based lighting, 317
- light position, 258
- lighting computation, 153
- main function, 82, 153
- Mandelbrot example, 471–472
- matrix functions, 138–139
- maximum number of texture image units, 119
- multitexturing example, 262–263
- noise, 400
- noise functions, 145–146
- noise to modify and animate shape, 418
- OpenGL Shading Language, 546
- output, 43
- output and debugging, 221
- output variables, 97
- passing texture coordinates from, 103
- per-vertex values, 80
- point size mode, 115–116
- position invariance, 117–118
- raster position, 117
- rendering with BRDF model, 373
- shader computation, 218
- shadow maps, 341–342
- soft volume shadow algorithm, 350, 351–352
- special built-in vertex shader output
 - variables, 97
- sphere morph, 415–417
- spherical harmonics lighting, 321, 337
- standard attributes, 100
- state, 46, 97
- technical illustration example, 466
- texture, 255–256
- texture access functions, 141–144
- texture maps, 99
- texturing, 118–120
- texturing example, 258–259
- toy ball, 294
- transforming incoming vertex position, 113
- trigonometry functions, 124–126
- two-sided color mode, 114–115
- überlight model, 325–326
- uniform variables, 99
- user clipping, 102, 116–117
- user-defined attribute variables, 168
- user-defined varying variable, 43
- varying qualified variables, 80
- varying variables, 43, 102, 163
- vector relational functions, 139–141
- Vertex-at-a-time interface, 184
- Vertex-at-a-time method, 9–10
- Vertices, 685
 - attributes, 14, 99–100, 184–195
 - completion of, 186
 - computing color, 242
 - data from client-side memory, 12
 - end of data definition, 10
 - eye coordinate position of incoming, 153
 - interpolating values, 80
 - normal, 152
 - parallel processing, 39
 - position, 101, 152
 - primary and secondary color, 13, 15
 - transforming and lighting single, 43



**Vertices, *continued***

- transforming position, 136, 138, 234
- values result of executing vertex shader, 98–99

Vidimice, Kiril, 323

View frustum, 686

View volume, 15, 686

Viewing direction, 153–154

- lighting, 240–241

- transforming, 302

Viewing matrix, 23, 686

Viewing parameters, 23

Viewing position, 154

Viewing transformation, 23, 686

Viewing vector, 156

Viewing volume, 25

Viewport mapping, 40

Viewport transformation, 26, 686

ViewPosition variable, 326

viewVec variable, 156

Visible light behavior, 364

Visual artifacts, 388

Visual complexity, 145–146

Visual Explanations (Tufte), 462

Void data types, 74

Volume shaders, 546

Volume shadow algorithm, 349–354

Volume shadows, 346–354

Voorhies, Douglas, 483

Voronoi mountainfractal, 509

Voronoi multifractal, 509

Voronoi noise, 508, 510–512, 517

W

Ward, Greg, 370–372

warn behavior, 92–93

Water ripples, 537

Wavelengths, 364–366

WCLightPos variable, 326

Weiblen, Mike, 227, 346

WGL routines, 6

while statement, 82

White light component colors, 364

Window coordinate system, 26, 687

Window coordinates, 15

Windowing system, 6

Windows, 5–7, 106

Wobble, 426–430

Wobble effect

- fragment shader, 429–430

- speeding up, 427

Wobble Tex variable, 427

Wood, 405–408

Wood shader application setup, 405–406

Woodcut print, 454

World coordinate system, 22, 687

World coordinates, 22–23, 326

World space, 22, 687

X

x raised to y power, 127

X Window System environment, 6

XML file format, 225

Xunitvec constant, 271

.xyz component selector, 295

Y

Yunitvec constant, 271

Z

Zoom factor, 20

