



NVIDIA Iray[®]

Release Notes

29 June 2023

Version 2023.0.1



Copyright Information

© 2023 NVIDIA Corporation. All rights reserved.

Document build number 367100.3452

Contents

1	Iray 2023.0.1, build 367100.3452	1
1.1	Added and Changed Features	1
1.1.1	General	1
1.1.2	Iray Photoreal & Iray Interactive	1
1.1.3	Iray Photoreal	1
1.2	Fixed Bugs	1
1.2.1	Iray Photoreal & Iray Interactive	1
1.2.2	Iray Photoreal	1
1.2.3	Iray Interactive	1
1.2.4	MI importer/exporter	1
2	Iray 2023.0.0, build 367100.2992	3
2.1	Added and Changed Features	3
2.1.1	General	3
2.1.2	Iray Photoreal	3
2.1.3	Material Definition Language (MDL)	3
2.2	Fixed Bugs	5
2.2.1	General	5
2.2.2	Iray API	5
2.2.3	Iray Photoreal & Iray Interactive	5
2.2.4	Iray Photoreal	5
2.2.5	Iray Interactive	6
2.2.6	Material Definition Language (MDL)	7
3	Iray 2023.0.0 beta, build 367100.1652	9
3.1	Known Issues and Restrictions	9
3.2	Added and Changed Features	9
3.2.1	General	9
3.2.2	Iray Photoreal & Iray Interactive	9
3.2.3	Iray Photoreal	9
3.2.4	Material Definition Language (MDL)	10
3.3	Fixed Bugs	13
3.3.1	General	13
3.3.2	Iray Photoreal & Iray Interactive	13
3.3.3	Iray Photoreal	13
3.3.4	Iray Interactive	14
3.3.5	Material Definition Language (MDL)	14
4	Iray 2022.1.9, build 363600.6589	15
4.1	Added and Changed Features	15
4.1.1	General	15

4.1.2	Iray Photoreal & Iray Interactive	15
5	Iray 2022.1.8, build 363600.5981	17
5.1	Fixed Bugs	17
5.1.1	Iray Photoreal	17
5.1.2	Iray Interactive	17
6	Iray 2022.1.7, build 363600.4887	19
6.1	Fixed Bugs	19
6.1.1	General	19
6.1.2	Iray Photoreal	19
6.1.3	Material Definition Language (MDL)	19
7	Iray 2022.1.6, build 363600.3938	21
7.1	Fixed Bugs	21
7.1.1	Iray Photoreal & Iray Interactive	21
7.1.2	Iray Photoreal	21
7.1.3	Material Definition Language (MDL)	21
8	Iray 2022.1.5, build 363600.3430	23
8.1	Added and Changed Features	23
8.1.1	General	23
8.1.2	Material Definition Language (MDL)	23
8.2	Fixed Bugs	23
8.2.1	General	23
8.2.2	Iray Photoreal	23
8.2.3	Iray Interactive	23
8.2.4	Material Definition Language (MDL)	23
9	Iray 2022.1.4, build 363600.2768	25
9.1	Added and Changed Features	25
9.1.1	General	25
9.2	Fixed Bugs	25
9.2.1	General	25
9.2.2	Iray Photoreal	25
9.2.3	Material Definition Language (MDL)	25
10	Iray 2022.1.3 internal, build 363600.2373	27
10.1	Fixed Bugs	27
10.1.1	General	27
10.1.2	Iray Photoreal	27
10.1.3	Material Definition Language (MDL)	27
10.1.4	MI importer/exporter	27

11 Iray 2022.1.2, build 363600.1913	29
11.1 Fixed Bugs	29
11.1.1 General	29
11.1.2 Iray Photoreal	29
12 Iray 2022.1.1, build 363600.1657	31
12.1 Fixed Bugs	31
12.1.1 General	31
12.1.2 Iray Photoreal	31
12.1.3 Material Definition Language (MDL)	31
13 Iray 2022.1.0, build 363600.1299	33
13.1 Known Issues and Restrictions	33
13.2 Added and Changed Features	33
13.2.1 General	33
13.2.2 Iray Photoreal & Iray Interactive	33
13.2.3 Iray Photoreal	33
13.2.4 Material Definition Language (MDL)	33
13.3 Fixed Bugs	33
13.3.1 General	33
13.3.2 Iray Photoreal & Iray Interactive	34
13.3.3 Iray Photoreal	34
13.3.4 Iray Interactive	35
13.3.5 Material Definition Language (MDL)	35
14 Iray 2022.1.0 beta, build 363600.482	37
14.1 Known Issues and Restrictions	37
14.2 Added and Changed Features	37
14.2.1 General	37
14.2.2 Iray API	37
14.2.3 Iray Photoreal & Iray Interactive	37
14.2.4 Iray Photoreal	38
14.2.5 Material Definition Language (MDL)	38
14.3 Fixed Bugs	39
14.3.1 General	39
14.3.2 Iray Photoreal & Iray Interactive	39
14.3.3 Iray Photoreal	39
14.3.4 Material Definition Language (MDL)	40
14.3.5 MI importer/exporter	40
15 Iray 2022.0.1, build 359000.3383	41
15.1 Known Issues and Restrictions	41
15.1.1 Material Definition Language (MDL)	41

15.2	Added and Changed Features	41
15.2.1	General	41
15.2.2	Material Definition Language (MDL)	41
15.3	Fixed Bugs	41
15.3.1	General	41
15.3.2	Iray Photoreal	41
15.3.3	Iray Interactive	42
15.3.4	Material Definition Language (MDL)	42
15.3.5	MI importer/exporter	42
16	Iray 2022.0.0, build 359000.2512	43
16.1	Known Issues and Restrictions	43
16.2	Added and Changed Features	43
16.2.1	General	43
16.2.2	Iray Photoreal & Iray Interactive	43
16.2.3	Material Definition Language (MDL)	43
16.3	Fixed Bugs	44
16.3.1	General	44
16.3.2	Iray Photoreal & Iray Interactive	44
16.3.3	Iray Photoreal	44
16.3.4	Iray Interactive	44
16.3.5	Material Definition Language (MDL)	45
16.3.6	MI importer/exporter	45
17	Iray 2022.0.0 beta, build 359000.876	47
17.1	Known Issues and Restrictions	47
17.2	Added and Changed Features	47
17.2.1	General	47
17.2.2	Iray Photoreal & Iray Interactive	47
17.2.3	Iray Photoreal	48
17.2.4	Material Definition Language (MDL)	48
17.3	Fixed Bugs	49
17.3.1	General	49
17.3.2	Iray Photoreal & Iray Interactive	49
17.3.3	Iray Photoreal	49
17.3.4	Material Definition Language (MDL)	50
17.3.5	MI importer/exporter	50
18	Iray 2021.1.7, build 349500.13092	51
18.1	Added and Changed Features	51
18.1.1	General	51
18.2	Fixed Bugs	51

18.2.1	General	51
18.2.2	Iray Photoreal	51
18.2.3	Iray Interactive	51
19	Iray 2021.1.6, build 349500.11420	53
19.1	Added and Changed Features	53
19.1.1	General	53
19.1.2	Material Definition Language (MDL)	53
19.2	Fixed Bugs	53
19.2.1	Iray Photoreal	53
19.2.2	MI importer/exporter	53
20	Iray 2021.1.5, build 349500.10431	55
20.1	Added and Changed Features	55
20.1.1	General	55
20.2	Fixed Bugs	55
20.2.1	Material Definition Language (MDL)	55
21	Iray 2021.1.4, build 349500.10153	57
21.1	Added and Changed Features	57
21.1.1	General	57
22	Iray 2021.1.3, build 349500.9894	59
22.1	Added and Changed Features	59
22.1.1	General	59
22.1.2	Material Definition Language (MDL)	59
22.2	Fixed Bugs	59
22.2.1	Iray Photoreal & Iray Interactive	59
22.2.2	Iray Photoreal	59
22.2.3	Iray Interactive	59
22.2.4	Material Definition Language (MDL)	60
22.2.5	MI importer/exporter	60
23	Iray 2021.1.2, build 349500.8766	61
23.1	Added and Changed Features	61
23.1.1	General	61
23.1.2	Iray Photoreal & Iray Interactive	61
23.1.3	Iray Photoreal	61
23.2	Fixed Bugs	61
23.2.1	Iray Photoreal & Iray Interactive	61
23.2.2	Iray Photoreal	61
23.2.3	Iray Interactive	61
23.2.4	Material Definition Language (MDL)	61

24 Iray 2021.1.1, build 349500.8264	63
24.1 Added and Changed Features	63
24.1.1 General	63
24.2 Fixed Bugs	63
24.2.1 General	63
24.2.2 Iray API	63
24.2.3 Iray Photoreal & Iray Interactive	63
24.2.4 Iray Photoreal	63
24.2.5 Iray Interactive	64
24.2.6 Material Definition Language (MDL)	64
24.2.7 MI importer/exporter	64
25 Iray 2021.1.0, build 349500.7063	67
25.1 Known Issues and Restrictions	67
25.2 Added and Changed Features	67
25.2.1 General	67
25.2.2 Iray API	67
25.2.3 Iray Photoreal & Iray Interactive	68
25.2.4 Material Definition Language (MDL)	68
25.2.5 MI importer/exporter	68
25.3 Fixed Bugs	68
25.3.1 General	68
25.3.2 Iray Photoreal & Iray Interactive	69
25.3.3 Iray Photoreal	69
25.3.4 Iray Interactive	70
25.3.5 Material Definition Language (MDL)	70
26 Iray 2021.1.0 beta, build 349500.5279	71
26.1 Known Issues and Restrictions	71
26.2 Added and Changed Features	71
26.2.1 General	71
26.2.2 Iray API	71
26.2.3 Iray Photoreal & Iray Interactive	72
26.2.4 Iray Photoreal	73
26.2.5 Iray Interactive	74
26.2.6 Material Definition Language (MDL)	74
26.3 Fixed Bugs	74
26.3.1 General	74
26.3.2 Iray API	74
26.3.3 Iray Photoreal & Iray Interactive	74
26.3.4 Iray Photoreal	75
26.3.5 Iray Interactive	75

26.3.6	Material Definition Language (MDL)	75
26.3.7	MI importer/exporter	76
27	Iray 2021.0.5, build 344800.12856	77
27.1	Fixed Bugs	77
27.1.1	General	77
27.1.2	Iray Photoreal & Iray Interactive	77
27.1.3	Iray Photoreal	77
28	Iray 2021.0.4, build 344800.9767	79
28.1	Fixed Bugs	79
28.1.1	General	79
28.1.2	Iray Photoreal & Iray Interactive	79
28.1.3	Iray Photoreal	79
28.1.4	Material Definition Language (MDL)	79
29	Iray 2021.0.3, build 344800.8726	81
29.1	Known Issues and Restrictions	81
29.2	Added and Changed Features	81
29.2.1	General	81
29.3	Fixed Bugs	81
29.3.1	Iray Photoreal & Iray Interactive	81
29.3.2	Iray Interactive	81
29.3.3	Material Definition Language (MDL)	81
30	Iray 2021.0.2, build 344800.7839	83
30.1	Added and Changed Features	83
30.1.1	General	83
30.1.2	Iray Photoreal & Iray Interactive	83
30.1.3	Iray Photoreal	83
30.1.4	Iray Interactive	83
30.1.5	MI importer/exporter	83
30.2	Fixed Bugs	83
30.2.1	General	83
30.2.2	Iray Photoreal & Iray Interactive	83
30.2.3	Iray Photoreal	83
30.2.4	Iray Interactive	84
30.2.5	Material Definition Language (MDL)	84
30.2.6	MI importer/exporter	84
31	Iray 2021.0.1, build 344800.4174	85
31.1	Added and Changed Features	85
31.1.1	General	85

31.1.2	Iray Photoreal & Iray Interactive	85
31.1.3	Iray Photoreal	85
31.1.4	Material Definition Language (MDL)	85
31.2	Fixed Bugs	85
31.2.1	General	85
31.2.2	Iray Photoreal & Iray Interactive	86
31.2.3	Iray Photoreal	86
31.2.4	Iray Interactive	86
31.2.5	Material Definition Language (MDL)	86
31.2.6	AIX importer	87
31.2.7	MI importer/exporter	87
31.2.8	Deep Learning based Denoiser	87
32	Iray 2021.0.0, build 344800.2052	89
32.1	Added and Changed Features	89
32.1.1	General	89
32.1.2	Iray API	89
32.1.3	Iray Photoreal	89
32.1.4	Material Definition Language (MDL)	89
32.2	Added and Changed Features	89
32.3	Fixed Bugs	91
32.3.1	General	91
32.3.2	Iray Photoreal	92
32.3.3	Iray Interactive	92
32.3.4	Material Definition Language (MDL)	92
32.3.5	MI importer/exporter	93
33	Iray 2021.0.0 beta, build 344800.351	95
33.1	Known Issues and Restrictions	95
33.2	Added and Changed Features	95
33.2.1	General	95
33.2.2	Iray API	96
33.2.3	Iray Photoreal & Iray Interactive	96
33.2.4	Iray Photoreal	97
33.2.5	Iray Interactive	97
33.2.6	Material Definition Language (MDL)	97
33.2.7	AxF importer	99
33.2.8	MI importer/exporter	99
33.2.9	Deep Learning based Denoiser	99
33.3	Fixed Bugs	100
33.3.1	General	100
33.3.2	Iray API	100

33.3.3	Iray Photoreal & Iray Interactive	100
33.3.4	Iray Photoreal	100
33.3.5	Iray Interactive	101
33.3.6	Material Definition Language (MDL)	102
33.3.7	AxF importer	103
33.3.8	MI importer/exporter	103
34	Iray 2020.1.6, build 334300.9558	105
34.1	Added and Changed Features	105
34.1.1	General	105
34.2	Fixed Bugs	105
34.2.1	Iray Photoreal & Iray Interactive	105
35	Iray 2020.1.5, build 334300.8936	107
35.1	Added and Changed Features	107
35.1.1	General	107
35.1.2	Iray Interactive	107
35.2	Fixed Bugs	107
35.2.1	Iray Photoreal & Iray Interactive	107
35.2.2	Iray Photoreal	107
35.2.3	Iray Interactive	107
35.2.4	Material Definition Language (MDL)	107
36	Iray 2020.1.4, build 334300.6885	109
36.1	Added and Changed Features	109
36.1.1	Iray Photoreal	109
36.2	Fixed Bugs	109
36.2.1	Iray Photoreal	109
37	Iray 2020.1.3, build 334300.6349	111
37.1	Added and Changed Features	111
37.1.1	General	111
37.1.2	Iray Photoreal	111
37.1.3	Material Definition Language (MDL)	111
37.2	Fixed Bugs	111
37.2.1	General	111
37.2.2	Iray API	111
37.2.3	Iray Photoreal & Iray Interactive	111
37.2.4	Iray Photoreal	111
37.2.5	Iray Interactive	112
37.2.6	Material Definition Language (MDL)	112
38	Iray 2020.1.2, build 334300.5582	113

38.1	Added and Changed Features	113
38.1.1	General	113
38.1.2	Material Definition Language (MDL)	113
38.2	Fixed Bugs	113
38.2.1	General	113
38.2.2	Iray Photoreal & Iray Interactive	113
38.2.3	Iray Photoreal	114
38.2.4	Iray Interactive	114
38.2.5	Material Definition Language (MDL)	114
38.2.6	MI importer/exporter	114
39	Iray 2020.1.1, build 334300.4226	115
39.1	Added and Changed Features	115
39.1.1	Material Definition Language (MDL)	115
39.2	Fixed Bugs	115
39.2.1	General	115
39.2.2	Iray API	115
39.2.3	Iray Photoreal & Iray Interactive	115
39.2.4	Iray Photoreal	115
39.2.5	Material Definition Language (MDL)	115
39.2.6	MI importer/exporter	116
40	Iray 2020.1.0, build 334300.2228	117
40.1	Known Issues and Restrictions	117
40.2	Added and Changed Features	117
40.2.1	General	117
40.2.2	Iray API	117
40.2.3	Material Definition Language (MDL)	117
40.2.4	Deep Learning based Denoiser	118
40.3	Fixed Bugs	118
40.3.1	General	118
40.3.2	Iray Photoreal & Iray Interactive	118
40.3.3	Iray Photoreal	118
40.3.4	Iray Interactive	119
40.3.5	Material Definition Language (MDL)	119
40.3.6	MI importer/exporter	120
40.3.7	Deep Learning based render progress	120
41	Iray 2020.1.0 beta, build 334300.1111	121
41.1	Known Issues and Restrictions	121
41.2	Added and Changed Features	121
41.2.1	General	121

41.2.2	Iray API	122
41.2.3	AxF importer	122
41.2.4	MI importer/exporter	122
41.2.5	Iray Photoreal & Iray Interactive	122
41.2.6	Iray Photoreal	123
41.2.7	Iray Interactive	123
41.2.8	Material Definition Language (MDL)	123
41.3	Fixed Bugs	124
41.3.1	General	124
41.3.2	MDL Compiler and Backends	124
41.3.3	Iray Photoreal & Iray Interactive	125
41.3.4	Iray Photoreal	125
41.3.5	Iray Interactive	126
41.3.6	MI importer/exporter	126
41.4	Iray 2020.0.3, build 327300.9514	126
41.5	Added and Changed Features	126
41.5.1	General	126
41.5.2	Iray Photoreal & Iray Interactive	126
41.6	Fixed Bugs	126
41.6.1	General	126
41.6.2	Iray Photoreal & Iray Interactive	127
41.6.3	Iray Photoreal	127
41.6.4	Iray Interactive	127
41.6.5	Material Definition Language (MDL)	127
42	Iray 2020.0.2, build 327300.6313	129
42.1	Added and Changed Features	129
42.1.1	General	129
42.1.2	Iray Interactive	129
42.1.3	Material Definition Language (MDL)	129
42.2	Fixed Bugs	129
42.2.1	General	129
42.2.2	Iray Photoreal	129
42.2.3	Iray Interactive	129
42.2.4	Material Definition Language (MDL)	129
42.2.5	Deep Learning based render progress - experimental feature	130
43	Iray 2020.0.1, build 327300.3640	131
43.1	Fixed Bugs	131
43.1.1	General	131
43.1.2	Iray Photoreal & Iray Interactive	131
43.1.3	Iray Photoreal	131

43.1.4	Iray Interactive	131
43.1.5	Material Definition Language (MDL)	131
44	Iray RTX 2020.0.0, build 327300.2022	133
44.1	Known Issues and Restrictions	133
44.2	Added and Changed Features	133
44.2.1	General	133
44.2.2	Iray API	133
44.2.3	Iray Photoreal	133
44.2.4	Material Definition Language (MDL)	133
44.3	Fixed Bugs	134
44.3.1	General	134
44.3.2	MDL Compiler and Backends	134
44.3.3	Iray API	134
44.3.4	Iray Photoreal & Iray Interactive	134
44.3.5	Iray Photoreal	134
44.3.6	Iray Interactive	135
44.3.7	Material Definition Language (MDL)	135
44.3.8	MI importer/exporter	135
44.3.9	Deep Learning based Denoiser	135
45	Iray 2020.0.0 beta, build 327300.312	137
45.1	Known Issues and Restrictions	137
45.2	Added and Changed Features	137
45.2.1	General	137
45.2.2	Iray API	137
45.2.3	MI importer/exporter	138
45.2.4	Iray Photoreal & Iray Interactive	139
45.2.5	Iray Photoreal	139
45.2.6	Iray Interactive	139
45.2.7	Material Definition Language (MDL)	139
45.2.8	Deep Learning based Denoiser	140
45.3	Fixed Bugs	140
45.3.1	MDL Compiler and Backends	140
45.3.2	Iray Photoreal & Iray Interactive	140
45.3.3	Iray Photoreal	140
45.3.4	Iray Interactive	140
45.3.5	Deep Learning based Denoiser	141
46	Iray 2019.1.8, build 317500.18465	143
46.1	Added and Changed Features	143
46.1.1	General	143

47 Iray 2019.1.7, build 317500.17646	145
47.1 Added and Changed Features	145
47.1.1 General	145
47.1.2 Iray Photoreal & Iray Interactive	145
48 Iray 2019.1.6, build 317500.11725	147
48.1 Added and Changed Features	147
48.1.1 Iray Photoreal	147
48.2 Fixed Bugs	147
48.2.1 MDL Compiler and Backends	147
48.2.2 Iray Photoreal & Iray Interactive	147
49 Iray 2019.1.5, build 317500.7473	149
49.1 Fixed Bugs	149
49.1.1 General	149
49.1.2 MDL Compiler and Backends	149
49.1.3 Iray Photoreal	149
49.1.4 Iray Interactive	149
50 Iray RTX 2019.1.4, build 317500.5529a	151
50.1 Added and Changed Features	151
50.1.1 General	151
50.1.2 Iray API	151
50.1.3 MI importer/exporter	151
50.1.4 Iray Photoreal	151
50.2 Fixed Bugs	151
50.2.1 Iray Photoreal & Iray Interactive	151
50.2.2 Iray Photoreal	151
50.2.3 Iray Interactive	152
50.2.4 MI importer/exporter	152
51 Iray RTX 2019.1.3, build 317500.3714	153
51.1 Fixed Bugs	153
51.1.1 General	153
51.1.2 Iray Photoreal & Iray Interactive	153
51.1.3 Iray Photoreal	153
51.1.4 Iray Interactive	153
52 Iray 2019.1.2, build 317500.2996	155
52.1 Fixed Bugs	155
52.1.1 MDL Compiler and Backends	155
52.1.2 Iray Photoreal & Iray Interactive	155
52.1.3 Iray Photoreal	155

52.1.4 Iray Interactive	155
52.1.5 Deep Learning based Denoiser	155

1 Iray 2023.0.1, build 367100.3452

1.1 Added and Changed Features

1.1.1 General

- Updated general libraries:
 - OpenSSL 1.1.1u
 - FFmpeg-lgpl-5.1.3-370316

1.1.2 Iray Photoreal & Iray Interactive

- Added missing support for Ada (SM8.9) and Hopper (SM9.0) on arm platforms.

1.1.3 Iray Photoreal

- Added motion vectors for the background/dome (i.e. no geometry hit) via the new `motion_vectors_background` option.

1.2 Fixed Bugs

1.2.1 Iray Photoreal & Iray Interactive

- Fixed normal flipping for tri-planar projection and adapt threshold in projector direction check to be more adaptive (nvbugs 3691327).

1.2.2 Iray Photoreal

- Improved handling of fibers/hairs if the caustic sampler is enabled.
- Fixed a double auxiliary buffer contribution when hitting volumes (nvbugs 4082804).
- Fixed ground fog not becoming uniform for identical density values and improved handling of invalid inputs (nvbugs 4138171).
- Fixed crashes in motion vector and uv auxiliary buffers when using inhomogeneous volumes (nvbugs 4150955).
- Changed ground fog spectral behavior, make it similar to the inhomogeneous behavior (nvbugs 4151165).

1.2.3 Iray Interactive

- Added rendering of the section caps for the `BSDF_weight` canvas (nvbugs 4131562).

1.2.4 MI importer/exporter

- Exporter: Adapt name optimization such that names are only dropped if the current name is in a sub-namespace of the current namespace (nvbugs 4147503).

2 Iray 2023.0.0, build 367100.2992

2.1 Added and Changed Features

2.1.1 General

- Updated general libraries:
 - OpenVDB 10.0.1-368791 (nvbugs 3957084)
 - SQLite 3.42.0

2.1.2 Iray Photoreal

- Added new approximative Mie phase function for volumes (via MDL 1.8), incl. ground fog (via `df::fog_vdf`).
- Performance of the new caustic sampler 2.0 has been vastly improved, depending on GPU and scene up to 2.1x!
- Performance of MDL-JIT-compiled materials has been vastly improved.
- Performance on Ada Lovelace generation (SM 8.9) has been improved.
- Performance in general (compared to both 2023.0.0 beta and partially 2022.X.X), especially for scenes containing fibers/hair and cutouts, has been improved.
- Optimized performance of rounded corners on pre-RTX generation GPUs.
- Implemented section caps for inhomogeneous volumes.

2.1.3 Material Definition Language (MDL)

- MDL 1.8 Language Specification
 - Updated version to 1.8.2 and removed the draft status of the document.
 - Removed meaningless references to a preprocessor.
 - Clarified that MDL comments can be nested.
 - Extend list of separators in MDL and clarify their purpose.
 - Clarified that *unicode identifiers* cannot be empty.
 - Fixed the accidentally dropped grammar production rule for the import statements in MDL files.
 - Clarified that let-bound variables cannot be used in their own initializer expressions.
 - Added a reference to the approximation used for the Mie scattering model of `fog_vdf`.
 - Clarified that the `particle_size` parameter of the new `fog_vdf` approximative Mie VDF model refers to the particle diameter.
- Deprecated `ILink_unit::add_environment()`. Use `ILink_unit::add_function()` with `ILink_unit::FEC_ENVIRONMENT` as second parameter instead. The old method is still available if `MI_NEURAYLIB_DEPRECATED_14_0` is defined.
- Improved documentation for the various subclasses of `IExpression`, in particular `IExpression_temporary` and `Expression_parameter`.
- The module builder supports now upgrades to MDL version 1.8, i.e., it expands namespace aliases to the corresponding Unicode identifiers, and finally removes the namespace aliases themselves.

- Added array support to the argument editor. New overloads of `Argument_editor::set_value()` and `get_value()` allow now to set or get entire arrays in one call.
- The `OpenImageIO` plugin for TIFF has been changed such that unassociated alpha mode is used for export. This avoids rounding errors and information loss for small alpha values.
- The MDL exporter now uses TIFF instead of EXR when exporting textures with an alpha channel. This avoids rounding errors and information loss for small alpha values.
- Various API methods related to resource loading have been fixed to properly report all error conditions. This required changing the existing error codes for those methods. The affected API methods are:
 - `mi::neuraylib::IImage::reset_file()`
 - `mi::neuraylib::IImage::reset_reader(IReader*, ...)`
 - `mi::neuraylib::IImage::reset_reader(IArray*, ...)`
 - `mi::neuraylib::ILightprofile::reset_file()`
 - `mi::neuraylib::ILightprofile::reset_reader()`
 - `mi::neuraylib::IBsdf_measurement::reset_file()`
 - `mi::neuraylib::IBsdf_measurement::reset_reader()`
 - `mi::neuraylib::IVolume_data::reset_file()`
 - `mi::neuraylib::IVolume_data::reset_reader()`
 - `mi::neuraylib::IMdl_factory::create_texture()`
 - `mi::neuraylib::IMdl_factory::create_light_profile()`
 - `mi::neuraylib::IMdl_factory::create_bsdf_measurement()`
 - `mi::neuraylib::IImport_api::import_bsdf_data()`
- Changed the default privacy level of `ITransaction::copy()` from `0` to `LOCAL_SCOPE` for consistency with `store()`. The old behavior is still available if `MI_NEURAYLIB_DEPRECATED_ITRANSACTION_COPY_DEFAULT_PRIVACY_LEVEL_ZERO` is defined.
- Fixed the VDB integration such that VDBs referenced in MDL archives or MDLEs work.
- Fixed `IMdl_factory::create_texture()` to support VDB files.
- Updated jquery to version 3.6.4 and/or applied workarounds to fix a potential XSS vulnerability.
- Replaced `ITarget_code::get_body_texture_count()` by `ITarget_code::get_texture_is_body_resource(mi::Size)` to query the information for the individual resources; same for light profiles and BSDF measurements.
- Allow to disable all example dependencies if the SDK and core examples are disabled.
- The finding python CMake script now falls back to the system installation if no path is specified and emits a warning in that case.
- Changed the Clang version and existence checks so that 12.0.1 is no longer a hard requirement.
- Disable LLVM CMake warning about `host=x64`.
- `IFunction_definition::get_mangled_name()` was added. For definitions representing real MDL functions this returns the Itanium ABI inspired mangled name used by backends when MDL code is compiled to target code.

- `IUnit::add_function()` was added, `IUnit::add_environment()` is now deprecated.
- `IUnit::add_function()` for function definitions added by 3ds.
- Changed the argument names of the array constructor. Name is now `value<n>` and no longer `<n>`. This might change expression paths to material compilation.
- `nvidia::core_definitions`
 - Texture transformations changed from uniform to varying.
 - Added triplanar texturing and blending of normals.
- Implemented normalization of `df::thin_film()` on the DAG.
- libbsdf: Avoid PDF code duplication caused by nested layerer DFs.
- Handle `\u`, `\U`, `\x`, and `\X` escapes more like C++
 - `\u` takes exactly 4 digits, it's an error if less are given.
 - `\U` takes exactly 8 digits, it's an error if less are given.
 - `\x` or `\X` without a digit is an error.
 - `\x` or `\X` with an overflow is a warning.
 - Illegal Unicode codepoints generated using `\u` or `\U` are now an error.
- Wrong UTF-8 encodings in MDL files are now an error.

2.2 Fixed Bugs

2.2.1 General

- Fixed issues with tessellation jobs.
- Resolved given path via path module when querying VDB file content.
- Prevented materials from being assigned as camera backplate and aperture shaders (nvbugs 200782051).

2.2.2 Iray API

- Fixed compilation issue on some combination of compiler and C++ standard for `iray_render_target.h`.

2.2.3 Iray Photoreal & Iray Interactive

- Fixed Hosek sky model sun limb darkening for RGB renderings.

2.2.4 Iray Photoreal

- Fixed termination criteria during light hierarchy construction, leading to a very rare crash (nvbugs 4100894).
- Fixed some crashes in guided sampling scheduling (includes nvbugs 4097747).
- Fixed CUDA crashes during rendering for scenes with texture compression enabled (nvbugs 4097747).
- Fixed `fiber_state::texture_coordinate(0).z` to contain thickness (i.e. diameter), not radius, matching the MDL spec.
- Improved behavior of the fiber/hair scattering model with caustic sampler enabled (nvbugs 4085079).

- Fixed the computation of motion vectors for fibers and particles.
- Improved spectral thin film code (MDL-1125).
- Fixed a crash for small resolutions on multiple devices (nvbugs 4099530).
- Fixed Light Path Expression crash in caustic sampler 2.0 if alpha was continued, but color already terminated.
- Fixed a crash for surface volume coefficients, in case only one is driven by a function (nvbugs 4127509).
- Fixed a regression if both rounded corners and caustic sampler are used, on CPU and older GPUs.
- Fixed a regression if both cutouts and caustic sampler are used (nvbugs 4068695).
- Fixed the false-color overlay of the finite sized environment when `environment_dome_visualize` is enabled (nvbugs 3815369).
- Fixed an issue with section caps to fix seemingly transparent clipping planes, this can also improve performance with section planes in general.
- Fixed beta regression with false matte ground hits if section caps enabled (nvbugs 4111804).
- Avoided ghost-like shadows when far-clipping inhomogeneous volumes and improved inhomogeneous volume clipping caps (nvbugs 4086715).
- Fixed ground fog ghost-like shadows (nvbugs 4082897).
- Fixed missing absorption within ground fog when far clipping outside the scene boundary.
- Fixed a rare numeric failure in ground fog rendering (nvbugs 4127423).
- Increased numerical robustness using ground fog in presence of extreme height differences (nvbug 4129471).
- Fixed beta regression regarding inhomogeneous volume rendering in combination with MDL JIT on the CPU.
- Slightly improved behavior in tight GPU memory situations.
- Fixed importance sampling of the `df::ward_geisler_moroder_bsdf` with multiscatter.
- Fixed multiple importance sampling numeric boundary case (nvbugs 4061585).
- Have CPU and CUDA log outputs match for caustic sampler 2.0 hierarchy construction.
- Saved a bit of memory for triangle geometry on RTX GPUs.
- Slightly improved pre-processing and update performance of GPU data.
- Slightly improved some corner cases that lead to more noise with the new caustic sampler enabled (nvbugs 4061578).

2.2.5 Iray Interactive

- Made Interactive behave more like Photoreal when shading lights in white mode (nvbugs 4056766).
- Clamp section cap colors (nvbugs 4085217).

2.2.6 Material Definition Language (MDL)

- Fixed endless loop in the MDL compiler that could happen if a preset is defined with some literals under care conditions.
- Fixed potential crash that could occur when an if and its else branch contain identical expressions.
- Fixed garbage issued in the "called object is not a function" error message.
- Fixed a crash when a single expression body function references an array length identifier.
- Fixed a crash in code generation when a MDL cast<> operator was used as an argument for an select instruction.
- Fixed non-working $\text{cast}\langle T \rangle(\text{cast}\langle S \rangle(x)) \implies \text{cast}\langle T \rangle(x)$ optimization.
- Fixed missing support for rvalues of constants in the JIT Backend.
- Fixed optimizer does not replace constants by literal values in loop initializers.
- Fixed DAG BE generating wrong signatures when struct insert operations are necessary.
- Fixed handling of global scope operator, $:x$ may now distinct from x .
- Printing now $\backslash U$ escape codes with 8 digits instead of 6 when exporting MDL.
- Fixed a race-condition in the JIT for the native backend leading to some error messages in cases with many CPU cores.
- The GLSL/HLSL backends now generate "Internal backend error" messages if some code could not be translated into the target language.
- Added support for some previously unsupported constructs in GLSL/HLSL.
- Fixed `opt_level 0` code generation for the PTX backend.
- Fixed alignment of vector, matrix, array, color and struct types in argument blocks for GLSL/HLSL.
- Improved error message when default constructing a struct which has unresolved field types.
- Fixed auto-generated name for single init function if none was provided. Previously duplicate symbol names were generated causing linker errors when using multiple materials with the PTX backend.
- libbsdf:
 - Fixed PDF of `df::backscattering_glossy_reflection_bsdf()`.
 - Fixed wrong multiscatter texture for `df::ward_geisler_moroder_bsdf()`.
 - Fixed importance sampling of `df::ward_geisler_moroder_bsdf` if `multiscatter` is on.
 - Fixed computation of diffuse part of pdf for all BSDFs with `multiscatter_tint`.
 - Fixed behavior of `df::thin_film` with thickness `0.0`.
 - Improved color correctness of `df::thin_film`.
- Fixed computation of pdf in measured BSDF in native runtime.

3 Iray 2023.0.0 beta, build 367100.1652

3.1 Known Issues and Restrictions

- Due to the new and improved caustic sampler, renderings that employ the caustic sampler are noticeably slower now. Hopefully, 2023.0.0 final will improve performance again.
- The new caustic sampler also does not feature the interactive scheduler yet. Thus, enabling the interactive scheduler with enabled caustic sampler will use the batch scheduler, but running at limited ramp up. 2023.0.0 final should also fix this behavior again.

3.2 Added and Changed Features

3.2.1 General

- Updated general libraries:
 - Boost 1.81
 - FFmpeg-lgpl-5.1.2-367645
 - OpenImageIO 2.4.9.0-367418
 - NVAPI R530
 - SQLite 3.41.2
 - libx264 (20210714-5db6aa6c-367481)
 - Curl 7.88.1
- Improved performance when detaching elements from a group. Iterating over the group using an index in order is still fast, but random access by index will be more costly and should be avoided.
- General further scene traversal/processing optimizations.

3.2.2 Iray Photoreal & Iray Interactive

- Add core support for the new MDL 1.8 features
 - Added support for "An Analytic BRDF for Materials with Spherical Lambertian Scatterers" (`df::dusty_diffuse_reflection_bsdf`).
 - Added support for `df::thin_film` modification of `df::custom_curve_layer` and `df::directional_factor`.

3.2.3 Iray Photoreal

- Vastly improved caustic sampler 2.0: It is now possible to simulate and collect more path contributions accurately, namely SDS (specular-diffuse/glossy-specular) segments, like found e.g. in swimming pools and mirror/glass interactions. This leads to a more accurate simulation, but comes at the price of increased rendering times (on the average) and a slight (around 100 MiB) memory overhead.
- Faster performance (up to 15 percent) for all long running renderings (e.g. 1000+ samples at FullHD, or 500+ samples at 4K resolution).
- Reduce binary size for the Iray Photoreal library by 40 percent.

3.2.4 Material Definition Language (MDL)

- MDL 1.8 Language Specification
 - Updated version to 1.8.
 - Reduced the requirement on thin-walled materials that they *should* have equal transmission from both sides.
 - Added part to supported selector string components for OpenEXR.
 - Added the operator function call to the precedence table of all operators.
 - Added the cast operator to the precedence table of all operators.
 - Moved operator from the set of words reserved for future use to the reserved words in use.
 - Added section for *unicode identifiers*.
 - Clarified that the use of variables in let-expressions is read-only.
 - Clarified that binary logical operators may or may not use short-circuit evaluation.
 - Added definition of lvalues and rvalues.
 - Removed redundant subsections that `light_profile` and `bsdf_measurement` have no members.
 - Added requirement to check for array out-of-bounds access and to return default-constructed values in this case.
 - Clarified that function parameters are lvalues in their procedural implementation bodies.
 - Added that function parameters can only be used as read-only rvalues in functions defined by an expression.
 - Added that function parameters can only be used as read-only rvalues in let-expression.
 - Changed the grammar for the let-expression from *unary_expression* to *assignment_expressions* for the expression affected by the variables. This ensures the right scope precedence, e.g., the let-expression ends at a sequence operator.
 - Removed using alias declarations. They are replaced by Unicode identifiers directly supported as package and module names.
 - Added operator function call syntax for all applicable operators.
 - Clarified that material parameters can only be used as read-only rvalues in material definitions.
 - Clarified that material parameters can only be used as read-only rvalues in let-expression.
 - Removed the support for annotations on packages with the `package.pkg` file.
 - Added Unicode identifiers for package and module names.
 - Changed productions for *import_path*, *qualified_import* to allow Unicode identifiers in package and module names.
 - Added a collapsed parameter to the `anno::in_group` standard annotation to control the UI presentation of groups.
 - Added the standard annotation `node_output_port_default` and the respective enumeration type `node_port_mode` to control the default initialization of output ports for external shade graph nodes when they are created for MDL functions.

- Added the standard annotation `native` to hint that a function or material might have a native implementation in the integration.
- Added the standard annotation `noinline` to hint that a function or material should not be inlined by the compiler.
- Added that texture spaces and their respective tangents return in volume shading their respective values at the surface entry point of the current light-transport path.
- Added that `state::texture_space` and the related tangent state functions return the zero vector in case the index is out of the range of available texture spaces.
- Clarified the exact interpolation curve for `math::smoothstep` and that it is undefined for a zero or reversed interpolation range.
- Added `float4x4` as supported data type for scene data lookup.
- Removed the requirement on the scene data lookup functions to provide literal strings as arguments for the scene data names. They are now uniform strings and can be connected to uniform string parameters of a material or function definition.
- Added a diffuse reflection BSDF modeling spherical Lambertian scatterers.
- Added an approximative Mie VDF model for fog and cloud-like media.
- Added `custom_curve_layer` and `directional_factor` as eligible base BSDF for the `thin_film` BSDF modifier.
- Clarified that the `measured_factor` modifier BSDF allows reflectivities above one and it is in the responsibility of a material author to ensure energy conservation.
- Added optional trailing commas to the productions of `parameter_list`, `argument_list`, `enum_type_declaration` and `annotation_block`.
- Added a new Appendix G on external bindings and a section for the recommended binding of MDL functions to nodes with multiple output ports.
- `base.mdl`: Added an implementation of `base::abbe_number_ior`.
- Improved precision of conversion of spectra to color, in particular for spectra with narrow peaks.
- Refactored `base::coordinate_projection()` to reduce size of generated GPU code.
- Added `mi::neuraylib::IMdl_backend::execute_init()` method to allow calling init functions in single-init mode for the native backend.
- `i18n` translation:
 - Added support for the new boolean parameter to the `in_group` annotation (collapsed).
 - Added support for legacy `"in_group$1.7"` annotation.
 - `i18n.exe` tool:
 - Issue warning when annotation is not supported.
 - Added support for the new added boolean parameter in the `in_group` annotation (collapsed).
 - Added support for legacy `"in_group$1.7"` annotation.
- The parameters of the MDL array constructor have been changed from `"0"`, `"1"`, etc. to `"value0"`, `"value1"`, etc. to match the MDL specification. This also affects paths to fields in the compiled material. Such paths are e.g. used by `ICompiled_material::lookup_sub_expression()`, `IMdl_distiller_api::create_baker()`, various methods on

`IMdl_backend` and the `Target_function_description` struct. Hardcoded paths need to be adapted. A frequently occurring pattern is `".components.<number>."` which needs to be changed to `".components.value<numner>."`.

- Removed unused enumerator value `DS_SEQUENCE` from `mi::neuraylib::IFunction_definition::Semantics`.
- Changed BMP import of alpha channels with `OpenImageIO`. While the correct behavior is debatable, it restores the historical behavior of `FreeImage`
- A selector parameter has been added to the methods `IImage::set_from_canvas()`, and `IImage_api::create_canvas_from_buffer()`, and `IImage_api::create_canvas_from_reader()`. The old signatures are deprecated. They are still available if `MI_NEURAYLIB_DEPRECATED_14_0` is defined.
- The signatures of `create_texture()`, `create_light_profile()`, and `create_bsdf_measurement()` on `IMdl_factory` have been changed: These methods take now a `IMdl_execution_context` pointer as last argument instead of the optional `mi::Sint32` pointer. The old signatures are deprecated. They are still available if `MI_NEURAYLIB_DEPRECATED_14_0` is defined.
- The method `IImage::create_mipmaps()` has been renamed to `IImage::create_mipmap()`. The old name is deprecated. It is still available if `MI_NEURAYLIB_DEPRECATED_14_0` is defined.
- It is now considered an error to apply the "A" selector to pixel types without alpha channel (instead of assuming a fake alpha channel with values 1.0f or 255).
- The interface `IImage_plugin` has been changed to allow selector support and better plugin selection (See the API reference documentation for details):
 - The method `supports_selectors()` has been added.
 - The method `open_for_reading()` has an additional parameter for the selector string.
 - The `test()` method takes now a reader instead of a buffer and the file size.
- The `OpenImageIO` plugin supports now also non-RGBA selectors. Right now this works only for `OpenEXR` textures, including `OpenEXR` multipart images (use the part name as selector or first selector component).
- Full Unicode support (including code points outside of the basic multilingual planes) for MDL source code and filenames.
- The "Tags" page on the admin HTTP server has a new option to show only tags currently begin pinned.
- The `FreeImage` plugin has been removed. All examples have been converted to use the `OpenImageIO` plugin instead of the `FreeImage` plugin.
- `libbsdf`: Added support for `df::dusty_diffuse_reflection_bsdf`.
- `libbsdf`: Added support for `df::thin_film` modification of `df::directional_factor` and `df::(color_)custom_curve_layer` (MDL-1048).
- Optimized ternary operators on `df::thin_film()` where the base BSDF is identical to the BSDF of the other case of the ternary.
- Added "enable_pdf" backend option to allow disabling code generation of PDF methods for distribution functions.

- MDL Core language support:
 - MDL 1.8 support is now complete and always enabled, no need for "mdl_next" option. Especially the following features were added:
 - `scene::lookup_data_float4x4()`.
 - `anno::native()`, `anno::node_output_port_default()`.
 - No lvalues inside let expressions and single expression functions are allowed.
 - Removed alias declarations.
 - "mdl_next" option is currently a no-op.
 - Improved some error messages.

3.3 Fixed Bugs

3.3.1 General

- Prevented leaf objects from reusing the tessellation of objects that have not been processed yet (nvbugs 4058103).
- Reduced pre-processing/compile time for displacement (nvbugs 4058217).
- Slightly changed outline thickness behavior (clamped line thickness to 1, changed default to 2) (nvbugs 4061343).

3.3.2 Iray Photoreal & Iray Interactive

- Fixed measured BRDFs never being evaluated on backsides.
- Fixed incorrect mono mode being used for builtin displacement texturing (nvbugs 4072494).

3.3.3 Iray Photoreal

- Fixed inverted behavior of the abbe number IOR computation, which will unfortunately change the look of a lot of scenes that feature dispersion.
- Fixed total internal reflection happening on coating if thickness is 0.0 via texture inputs.
- Excluded environment-behind-volume hits from auxiliary buffers (nvbugs 4018115).
- Removed implicit multiplication with `meters_per_scene_unit` in MDL-JIT compiled volume materials, and in general fix several issues related to this.
- Replaced self intersection handling code once more, in order to get some previously lost performance back.
- Fixed rendering selection subset buffers in presence of decals, and other incorrect contributions when handling decals in auxiliary buffers (nvbug 3976531).
- Fixed matte and ground fog attenuation interactions (nvbugs 4045924).
- Improved numerical robustness of ground fog in case of a strong difference between the color channel values.
- Fixed a numerical error in ground fog (nvbugs 4070589).
- Made matte and ground fog compatible with section planes (nvbugs 4052312).
- Fixed auxiliary buffers featuring ground fog (nvbugs 3983116,4035932).

- Fixed crash if scene data lookups with names provided via MDL material parameters are used.
- Fixed adding multiple instances of the same particle object, leading to crashes (nvbugs 4061583).
- Fixed missing auxiliary buffer values if paths were terminated before the first material interaction (nvbugs 4035932).
- Enforced linear interpolation for texture runtime in JIT-compiled environment functions to increase the compatibility with the builtin envmap lookup and to reduce banding observed in some sky shaders.
- Fixed potential crashes with particles and cutouts.

3.3.4 Iray Interactive

- Behave more like Photoreal when shading lights in white mode.

3.3.5 Material Definition Language (MDL)

- Mark BSDF and EDF data structures as 16-byte aligned to avoid a crash due to misaligned stores in generated CPU code.
- Fixed IES parser for files with IES:LM-63-2019 standard.
- Fixed the export of textures inside .mi namespaces.
- MDL Core: Due to an inconsistency in the MDL Specification the MDL compiler was parsing erroneously some types of expressions after the keyword 'in' in the 'let' expressions, leading to a compiler error and resulting in that only unary type of expressions could be parsed correctly.
- libbsdf:
 - Fixed the handling of total internal reflection for microfacet BSDFs evaluation in `df::scatter_transmit` mode.
 - Correctly handle `df::thin_film` with a thickness of zero, independent of the IOR value.
- Fixed rare crash in HLSL code generator regarding translation of load instructions.
- Fixed missing module cache usage for non-inlined functions for link units.
- Fixed special `df::thin_film()` handling for PTX backend.
- Fixed crash for zero sized arrays in argument block layout builder.
- Fixed inlining of functions with deferred-size arrays as parameters.
- Fixed several bugs in the MDL compiler when compiling syntactically invalid MDL code.
- Fixed bug in the attribute handling of the Distiller.

4 Iray 2022.1.9, build 363600.6589

4.1 Added and Changed Features

4.1.1 General

- Updated general libraries:
 - OpenSSL 1.1.1u
 - SQLite 3.42.0

4.1.2 Iray Photoreal & Iray Interactive

- Added missing support for Ada (SM8.9) and Hopper (SM9.0) on arm platforms.

5 Iray 2022.1.8, build 363600.5981

5.1 Fixed Bugs

5.1.1 Iray Photoreal

- Fixed termination criteria during light hierarchy construction, leading to a very rare crash (nvbugs 4100894).

5.1.2 Iray Interactive

- Made Interactive behave more like Photoreal when shading lights in white mode (nvbugs 4056766).

6 Iray 2022.1.7, build 363600.4887

6.1 Fixed Bugs

6.1.1 General

- Prevent leaf objects from reusing the tessellation of objects that have not been processed yet (nvbug 4058103).

6.1.2 Iray Photoreal

- Fixed total internal reflection happening on coating if thickness is 0.0 via texture inputs.
- Fixed rendering selection subset buffers in presence of decals (nvbug 3976531).
- Fixed adding multiple instances of the same particle object, leading to crashes (nvbugs 4061583).

6.1.3 Material Definition Language (MDL)

- Fixed rare crash in HLSL code generator regarding translation of load instructions.
- Fixed crash for zero sized arrays in argument block layout builder.
- Several bugs fixed in the MDL compiler when compiling syntactically invalid MDL code.

7 Iray 2022.1.6, build 363600.3938

7.1 Fixed Bugs

7.1.1 Iray Photoreal & Iray Interactive

- Fixed a crash when constants are used as layer normals in MDL.

7.1.2 Iray Photoreal

- Fixed Ambient Occlusion kernel overwriting the hit state, leading to GPU failures or CPU crashes (nvbugs 4020134).
- Fixed missing change detection when a new material is assigned to a volume (nvbugs 4012311).
- Fixed rendering selection subset buffers on RTX cards (nvbugs 3891603).
- Fixed missing ground fog absorption when paths miss the scene bounding box (nvbugs 3983116).
- Improved the interaction of ground fog if a finite dome is used.

7.1.3 Material Definition Language (MDL)

- Fixed layout of API reference documentation for types using the `__align__(x)` macro.
- Fixed `ITargetCode` serialization for cases when there are multiple materials in the link unit.
- Fixed some cases where invalid MDL source code can lead to compiler crashes.
- Fixed total internal reflection happening on coating if thickness is 0.0 (libbsdf).

8 Iray 2022.1.5, build 363600.3430

8.1 Added and Changed Features

8.1.1 General

- Improved robustness when handling CUDA render targets supplied by the integration.
- Improved performance of texture import via the OpenImageIO plugin, in particular when only the metadata is needed.

8.1.2 Material Definition Language (MDL)

- Improved output messages for errors in annotations.

8.2 Fixed Bugs

8.2.1 General

- Fixed crashes with Iray Server (nvbugs 3875764).
- Slightly improved OpenGL error handling.

8.2.2 Iray Photoreal

- Fixed a crash with guided sampling, if the caustic sampler was toggled while guidance remained active (nvbugs 4011777).
- Fixed issues when changing a material to become inhomogeneous participating media (and other corner cases) (nvbugs 3972155).
- Fixed artifacts when rendering with guided sampling on multiple devices (nvbugs 3978162).
- Fixed invalid auxiliary buffer values when clipping inhomogeneous volumes (nvbugs 3970624).
- Hopefully prevent launch failures due to previously unknown launch memory requirements when using guidance.

8.2.3 Iray Interactive

- Fixed occasional pixel garbage when rendering frame subwindows (nvbugs 3964935).

8.2.4 Material Definition Language (MDL)

- Fixed potential crashes in the code generator when a module name contains a `.`.
- Fixed crash in the MDL compiler caused by invalid constant declarations.

9 Iray 2022.1.4, build 363600.2768

9.1 Added and Changed Features

9.1.1 General

- Updated general libraries:
 - JsonCpp 1.9.5
 - OpenImageIO 2.4.5.0 (without dbghlp dependency)
 - OpenSSL 1.1.1t
- The priority of the FreeImage and OpenImageIO plugin has been decreased by one level, such that it is possible again to override the default plugin with a custom image plugin.

9.2 Fixed Bugs

9.2.1 General

- Improved performance of scene updates, especially visible for huge scenes.
- Added missing retrieval of per-vertex userdata attributes of on demand meshes during scene updates.

9.2.2 Iray Photoreal

- Fixed crash when using uniform userdata for volumes.
- Improved navigation speed in large scenes with enabled motion blur, but no (or rare) motion.
- Fixed a rare crash caused by missing cleanups of device data for MDL-JIT-compiled functions.

9.2.3 Material Definition Language (MDL)

- Fixed a crash in the MDL compiler if a variant of a struct constructor is created.
- Fixed calculation of return type for variants of auto declared functions.
- Allow the creation of variants of the material type itself.
- Fixed wrong optimization of the body of function variants.
- Fixed crash when processing invalid annotations.
- Fixed handling of struct fields with incomplete type.
- Fixed crash when the same namespace is declared again.
- Fixed negative results in `df::fresnel_factor` in numerical corner cases (libbsdf).

10 Iray 2022.1.3 internal, build 363600.2373

10.1 Fixed Bugs

10.1.1 General

- Fixed loading of uv-tilsets/ animated textures with filenames containing meta-characters of regular expressions.

10.1.2 Iray Photoreal

- Fixed incorrect texture list indices for builtin bumps if used as dependency for MDL-JIT-compiled code, e.g. in case a normal input slot of a builtin bump function was connected to another function, the input slot was not consumed.
- Fixed spectral importance sampling weighting in case the scattering coefficient is zero in one channel.
- Changed the way internal ray tracing API contexts are managed. Contexts are now shared more broadly, which avoids re-initialization in some use cases.
- Fixed the instantaneous shutter mechanism for motion vectors. It also makes sure that the alpha and albedo buffers behave like the beauty buffer when the instantaneous shutter flag is on and that the shutter offset and the instantaneous shutter have no effect if motion vectors are disabled.
- Avoid computing camera motion blur if motion vectors instantaneous shutter is on and motion vectors are enabled.

10.1.3 Material Definition Language (MDL)

- Fixed a crash related to parameters declared with auto type.
- Fixed wrong compilation and/or possible crashes if a function with an assignment of an atomic type to a vector type was inlined into a material body.
- Fixed generated code for MDL math functions that are directly mapped to HLSL/GLSL intrinsics when several overloads of the same intrinsic are used.
- Fixed uninitialized variables in generated HLSL code in some cases when the layer and the base BSDF of a layering distribution function are the same.
- Fixed support for `modf()` and `sincos()` for HLSL and GLSL.

10.1.4 MI importer/exporter

- Fixed a crash during the export of on-demand meshes with userdata.

11 Iray 2022.1.2, build 363600.1913

11.1 Fixed Bugs

11.1.1 General

- Fixed sphere/particles bounding box computation.
- Fixed sphere/particles motion vectors to get lost when editing a particle object.

11.1.2 Iray Photoreal

- Fixed a problem with guided sampling where a large previous iteration batch before a camera move caused only single iteration updates from there on.
- Fixed guided sampling iteration scheduling overshooting the number of iterations between updates.
- Fixed partial framebuffer rendering support (needed to fit very very large framebuffers onto smaller GPU models).
- Fixed tint loss if different VDFs with different colors are mixed for spectral rendering.
- Fixed a crash with spheres/particles in case the (triangle or fiber) instance count does not match the object count (on RTX/Turing and up GPUs).

12 Iray 2022.1.1, build 363600.1657

12.1 Fixed Bugs

12.1.1 General

- Fixed a memory leak regression in image canvas introduced with the 2022.0.0 beta.
- Fixed multiple small issues with the new multi matte canvas.

12.1.2 Iray Photoreal

- Fixed a missing material update for MDL JIT-compiled materials when changing the spectral rendering parameter (nvbugs 3887626).
- Fixed the new selection flag for CPU/Embree spheres/particles and fibers (nvbugs 3891574).
- Fixed a wrong condition for (indirect) ghostlight detection when using the caustic sampler.
- Fixed block artifacts in ghostlights when also using guided sampling (nvbugs 3906572).
- Fixed way too large temporary memory buffer usage on pre-RTX/pre-Turing GPUs when using spheres/particles or fibers.
- Fixed an efficiency regression issue when rendering moderate or small resolutions, if guided sampling is enabled (as the scheduler did not crank up step sizes like it still did in the Iray 2022.1.0 beta) (nvbugs 3906561).

12.1.3 Material Definition Language (MDL)

- Restored compatibility of `base::perlin_noise_bump_texture()`, `base::worley_noise_bump_texture()`, and `base::flow_noise_bump_texture()` with previous versions (differences were noticeable, in particular with strong bump factors). Note that the server component of the Iray Bridge needs to be updated as well to obtain correct renderings (although the Bridge compatibility with Iray 2022.1.0 itself is not affected).

13 Iray 2022.1.0, build 363600.1299

13.1 Known Issues and Restrictions

- Newer driver versions (R525 branch) fix a brightness/color clamping issue when using the AI denoiser, especially with bright images (nvbugs 3760304).
- Also due to additional performance improvements, R525 GA2 (Windows: 526.67, Linux: 525.53) is thus the recommended driver for Iray 2022.1.0.
- Note that due to the new features listed below, this time unfortunately the bridge had to be changed compared to the 2022.1.0 beta.

13.2 Added and Changed Features

13.2.1 General

- Replaced still leftover FreeImage usages by OpenImageIO (e.g. SDK examples, Server, plugins).
- Updated general libraries:
 - OpenSSL 1.1.1s
 - OpenImageIO 2.4.5.0

13.2.2 Iray Photoreal & Iray Interactive

- Added multi matte canvas. This adds a new render target type (see `mi::neuraylib::TYPE_MULTI_MATTE`) and accompanying parameter type to the public API. It offers a scalar or vector canvas that contains one coverage mask per channel.

13.2.3 Iray Photoreal

- Native ghostlight support, a new concept to make lights seemingly invisible when viewed via specular or glossy reflection, see e.g. `ghostlight_factor` for how it is used, or the new section in the programmers manual (IRAY-1753).
- Added LPE support for emissive volumes and introduced a new light subtype, `Lv`, to denote light emitting volumes in LPEs (nvbugs 3867389).
- Added UV texture coordinate computation for spheres/particles.
- Added support for spheres/particle motion blur.

13.2.4 Material Definition Language (MDL)

- Improved Windows implementation of `mi::base::Lock` to use `SRWLOCK` instead of `CRITICAL_SECTION` primitives.

13.3 Fixed Bugs

13.3.1 General

- Fixed releasing of scopes (nvbugs 3877982).
- Support gray-alpha images with 16 and 32 bits per channel in OpenImageIO plugin (nvbugs 3829963).

- Fixed missing initialization of the `volume_priority` attribute.
- Fixed a bug where `nvenc_encoder` failed to load on a host with no GPU driver, causing the worker to refuse to start (nvbugs 3874131).
- Fixed missing tracking of changed geometry status when adding spheres/particles or setting particle data.
- Re-enabled network timeout by default, which was disabled by accident in an earlier release. Without the timeout, hosts leaving the cluster were not detected in UDP mode.

13.3.2 Iray Photoreal & Iray Interactive

- Fixed ground/matte shadow for a sun disk size of 0 (nvbugs 3867412).

13.3.3 Iray Photoreal

- Made guided sampling fully deterministic, so renderings on the same rendering setup will result in exactly the same image.
- Fixed spectral rendering accidentally clamping EDF values.
- Fixed more issues with and added some small optimizations for the optional instance camera shift (nvbugs 3860755).
- Fixed some performance regressions for materials featuring glossy transmission with no cutouts.
- Fixed missed intersections on transformed linear fibers (nvbugs 3845215).
- Fixed an issue with guided EDF sampling and direct light sampling (nvbugs 3858634).
- Fixed near clipping events to enable volumetric properties (nvbugs 3787726).
- Fixed remaining triangle/self intersection precision issues on Turing-without-RTcores (GeForce GTX and low-end mobiles with sm75), A100 (sm80) and H100 (sm90) (nvbugs 3870551 and 3881264).
- Fixed some remaining triangle/self intersection precision issues on CPU (nvbugs 3883738).
- Retweaked some ambient occlusion and rounded corners precision issues (nvbugs 3864452).
- Fixed crashes happening when using the ground/global fog (nvbugs 3872355).
- Fixed an issue with volumes and far clipping planes (nvbugs 3878355).
- Fixed a crash when rendering AUX buffers in combination with inhomogeneous volumes.
- Fixed numerical corner cases where the hair BSDF could create invalid numbers (Jira MDL-798).
- Made volume emission vs scattering/background scale independent.
- Improved section objects/caps interaction with SSS/volumes (nvbugs 3878246).
- Avoided some severe performance overhead (around 1.2x, depending on rendering setup and scene) for enclosing glossy transmission-only hulls, like found in a popular public benchmark scene (nvbugs 3846081).
- Optimized light hierarchy traversal, especially noticeable on CPU.

- Optimized hair BSDF component selection to reduce variance in importance sampling (OM-72862). Note that this will also lead to noticeably brighter hair rendering in some cases, if the firefly filter is enabled.
- Further reduced memory usage for volume data a bit, especially visible for huge, but rather sparse volumes.

13.3.4 Iray Interactive

- Fixed an issue with sampling the glossy component for environment maps/IBL (nvbugs 3557365).

13.3.5 Material Definition Language (MDL)

- libbsdf: Optimized hair BSDF component selection to reduce variance in importance sampling (OM-72862).
- Fixed conversion of spectral to color for the case that the minimum wavelength is greater than 380 nanometers.
- Fixed `IType_factory::is_compatible()` for enums (the sets of enumeration values need to be equal, not just one a subset of the other).
- OpenImageIO plugin
 - Fixed import of gray-alpha images with 16 and 32 bits per channel.
- Fixed crash inside the MDL core compiler when an enum value is used as the right hand side of a select expression.
- Fixed crash in the MDL core compiler when the qualified name of an import declaration is ill formed.
- Changed the behavior of state functions when used with an invalid texture space. Now they return always zero.

14 Iray 2022.1.0 beta, build 363600.482

14.1 Known Issues and Restrictions

- The minimum NVIDIA driver branch version in order to support the new CUDA and OptiX features is R520 (Windows 522.06, Linux 520.61.05, and up).
- FreeImage is now removed, due to its missing source code maintenance. It is completely replaced now by the OpenImageIO plugin. All examples were also switched over now.
- Removed `get/set_backplate_dof_enabled`, as it was already deprecated in 2015.
- Removed the Blend renderer.

14.2 Added and Changed Features

14.2.1 General

- This release adds native support for all Ada Lovelace and Hopper generation GPUs. Note though that all Lovelace GPUs are already supported implicitly since the Iray 2020.1.0 Beta (build 334300.1111).
- Updated general libraries:
 - CUDA 11.8
 - OptiX 7.5 (which addresses (at least) large parts of nvbugs 200777017, caused by AI denoiser temporary memory)
 - zlib 1.2.13.patched-363203
 - FFmpeg-lgpl-5.1.2-362780
 - SQLite 3.39.4
 - OpenImageIO 2.4.2.1-dev
- Added support for loading volume data via `IReader`.
- Added support for CUDA render targets to the bridge render context.

14.2.2 Iray API

- Added passing of fiber data via arrays rather than one by one.

14.2.3 Iray Photoreal & Iray Interactive

- Implemented new Hosek-Wilkie spectral sky model.
- Added support for a world space hit point auxiliary/AOV buffer.
- Added support for a new render target buffer producing outlines around selected objects. In addition to the new canvas type and associated parameters, this introduces a new boolean flag, `selected` to control which scene elements are included in the auxiliary buffers to drive the outline generation, i.e. which elements receive an outline. In order to align with the outline feature, the toon buffers edge color now features a fourth component to control alpha blending between the outlines/toon edges and the beauty image. Note that there is no compatibility fallback, so integrations must be adapted to supply the correct parameter type.

14.2.4 Iray Photoreal

- Guided sampling now keeps and updates the stored guidance data during camera movement/updates (instead of resetting it). This greatly enhances interactive rendering quality/convergence for complex scenes (e.g. indoor/architectural).
- Guided sampling is now also implemented for the caustic sampler, which increases its efficiency for complex scenes, especially ones which are spatially large or with a lot of multi-bounce caustic paths.
- Due to the latter improvement, enabling the caustic sampler will now also always force-enable guided sampling.
- Added support for new sphere/particle primitive to help with rendering particle systems and the like. This is exposed via `IParticles`. A new example has also been added.
- Added support for per-vertex fiber motion blur.
- Added support for linear segment fibers (which will render faster than the existing higher order fibers/curves).
- Added support for Catmull-Rom fiber data.
- Added parallel building of RTX acceleration structures (e.g. scenes with many instances will now preprocess faster on RTX GPUs).
- Added explicit control of grazing angle reflectivity of ground reflections via `environment_dome_ground_reflectivity_grazing` (default = -1, which is interpreted as white but also keeps the behavior of disabling ground reflections if `environment_dome_ground_reflectivity_grazing` is less or equal to zero) (nvbugs 3760308).
- JIT compiled MDL code now supports incremental updates, leading to less recompilation on sparse material updates.
- JIT compiled MDL environment and backplate functions are evaluated in the rendering core now and are no longer baked upfront, leading to faster updates/pre-processing, and no more fidelity loss of procedural environments/backplates. Backplate functions are now inversely tonemapped in core. Note that this may change the appearance of environments and backplates which were previously baked. (nvbugs 3785159)
- Added warnings if MDL generated code (for run time evaluation, material, environment, and backplates) converts spectra to RGB, if spectral rendering is enabled.

14.2.5 Material Definition Language (MDL)

- Allowed varying values on parameters of `base::rotation_translation_scale()`.
- Added the context option "export_resources_with_module_prefix" (defaults to true). If set to false, the module prefix is avoided if resources are exported as part of an exported MDL module.
- Python Bindings
 - Added high-level Python binding module `pymdl.py`.
 - Generated `.sh` and `.bat` scripts to run Python examples without manually setting `PATH` and `PYTHONPATH`.
- Added support for Unicode identifiers to the MDL compiler (for MDL version ≥ 1.8).

- Added implementations for `state::geometry_tangent_u()` and `state_geometry_tangent_v()` functions in JIT generated code. Before they were returning `0`.
- Distilling
 - Removed built-in target `diffuse_glossy`.

14.3 Fixed Bugs

14.3.1 General

- Changed BMP export of alpha channels with OpenImageIO. While the correct behavior is debatable, it restores the historical behavior of FreeImage (nvbugs 3853252).
- Changed the compression level for the export of PNG files back to 5 for much a better speed/space tradeoff (nvbugs 3647089).
- Fixed handling of hostnames starting with a number, e.g. `123.com`.

14.3.2 Iray Photoreal & Iray Interactive

- Protect Perez sun and sky model against implausible input (nvbugs 3447755).

14.3.3 Iray Photoreal

- Drastically reduced memory usage when rendering inhomogeneous volumes. At the same time this means that the trilinear volume filtering had to be changed. It now works on the MDL function, rather than the input data, which usually outputs an overall smoother result.
- All state is now available to backplate functions on backplate meshes as they are no longer baked (e.g. previously the `W` texture coordinate was ignored) (nvbugs 3741312).
- Fixed wrong computation of bounding boxes for rendering of fibers on pre-RTX/pre-Turing GPUs, leading to wrong rendering on some transformations.
- Fixed an issue with active instancing and multiple scopes which could lead to ghosting artifacts (nvbugs 3843793).
- Fixed incorrect cutout evaluation leading to a semi-transparent surface in case a simple JIT-compiled MDL expression drives the cutout.
- Spectral volume coefficients were converted to RGB even if with spectral rendering was enabled. This has been fixed (nvbugs 3818865).
- Made lightpath connection handling of rough materials with same IOR on both sides consistent/more correct.
- Improved robustness and convergence behavior of guided sampling, including support for SSS/homogeneous volumes.
- Changed behavior of auxiliary/AOV buffers when rendering with depth of field (single sample images now are basically ignoring DOF).
- Further improved self intersection handling/precision on all setups (CPU/pre-RTX/RTX) (nvbugs 3634421, 33756731, 634421, 2839115, 2869227, 2884158, 2948029, 2948171, 3296260, 3621145, 3625574, 3632810, 3632823).
- Improved performance of fiber rendering on RTX cards (at the price of slightly more memory usage).

14.3.4 Material Definition Language (MDL)

- Fixed artifacts in `base::perlin_noise_bump_texture()`, `base::worley_noise_bump_texture()` and `base::flow_noise_bump_texture()` when using small cell sizes.
- Fixed `IFunction_call::reset_argument()` for cases where a default parameter referenced the directly preceding parameter.
- Fixed `IFunction_call::set_argument()` to reject expressions that contain (forbidden) nested parameter references. Similarly for `IFunction_definition::create_call()`.

14.3.5 MI importer/exporter

- Fixed broken export of triangle mesh motion vectors in case more than one motion step exists.
- Fixed reading all triangle mesh motion steps, not just the first one.

15 Iray 2022.0.1, build 359000.3383

15.1 Known Issues and Restrictions

15.1.1 Material Definition Language (MDL)

- Adding an environment function to a link unit (via method `mi::neuraylib::ILink_unit::add_environment()`) is currently broken for all backends and will result in a wrong code. As a work-around, please generate environment functions as single code (i.e. do not add it to any link unit)

15.2 Added and Changed Features

15.2.1 General

- Updated general libraries:
 - FreeImage-3.19.x-r1903-OpenEXR-3.1.5-libTIFF-4.4.0-361926 (fixes problems/crashes with huge TIFF metadata `TIFFTAG_RICHTIFFIPTC`)

15.2.2 Material Definition Language (MDL)

- Added compiler context option "mdl_next": Setting this option will enable preliminary features from the upcoming MDL 1.8 Version, especially:
 - Full utf8 identifiers in MDL.
 - Lifted restriction on scene access functions: The name of a scene data can be now any expression of type uniform string.
- Python bindings
 - Updated the SWIG file in order to support `pymdlsdk`. `IFunction_definition.get_mdl_version()` in Python.
 - Handle `Mdl_version` as output parameter. `(since, removed) = mdl_func.get_mdl_version()`.

15.3 Fixed Bugs

15.3.1 General

- Fixed failed worker renderings with error invalid parameters (nvbugs 3759010).

15.3.2 Iray Photoreal

- Fixed a CPU crash with guided sampling enabled (nvbugs 3758281).
- Fixed some issues with guided sampling in combination with the caustic sampler or SSS.
- Optimized fiber rendering performance on Turing and above (1.1x to 1.7x, at the price of a bit more memory).
- Fixed/Unify ambient occlusion rays to be able to hit invisible-to-primary-rays geometry on all GPU/CPU configurations.

15.3.3 Iray Interactive

- Fixed slight precision problems with Oren-Nayar BSDFs (nvbugs 3777916).

15.3.4 Material Definition Language (MDL)

- Fixed error message regarding wrong array type issued two times.
- Speed up material compilation: Instantiation of a material instance is now faster due to less database queries and lesser reference counted operations.
- Fixed module inliner handling of relative resource paths.
- Fixed libbsdf handling of Fresnel factors. (OM-52209)

15.3.5 MI importer/exporter

- Fixed parsing of attributes of image textures.

16 Iray 2022.0.0, build 359000.2512

16.1 Known Issues and Restrictions

- The minimum NVIDIA driver version in order to support the new CUDA and OptiX features is R510 U6 (Windows 512.78, Linux 510.73.05). Note that the R515 driver series unfortunately features a performance regression that only affects Iray 2022.0.0. So when using R515 drivers, only drivers 516.93 (Windows), 515.65.01 (Linux) and up feature the necessary performance fix (nvidia 3707149 and related to 3678591).
- On Linux and macOS/x86 SSE3 enabled CPUs are now the new minimum requirement.
- Removed support for the already deprecated portal lights in Iray Photoreal. It is now suggested to use the new guided sampling and/or the improved caustic sampler instead.
- FreeImage is now deprecated, due to its missing source code maintenance. It will be replaced by the now also included OpenImageIO plugin. Please test this evaluation build and let us know if this new plugin features regressions compared to FreeImage, or worse performance.

16.2 Added and Changed Features

16.2.1 General

- Add versioning to the IndeX Direct libraries.
- Updated general libraries:
 - OpenSSL 1.1.1q
 - FreeImage-3.19.x-r1903-OpenEXR-3.1.5-libTIFF-4.4.0-360897
 - FFmpeg-lgpl-4.4.2-360169
 - OpenEXR 3.1.5
 - Embree-3.13.2-359470 (with a custom patch for the crashes seen in nvidia 3643565)
 - AxF 1.9.0 (now supports all platforms)
 - IndeX Direct 359000.2512 (fixes an issue with empty VDBs, nvidia 3720395)
 - OpenImageIO 2.4.1.2 (for evaluation purposes)

16.2.2 Iray Photoreal & Iray Interactive

- Added support for `df::thin_film` modifier for glossy BSDFs that feature mode `df::scatter_reflect_transmit` (OM-52209).

16.2.3 Material Definition Language (MDL)

- Added contribution instructions and a Contributor License Agreement (CLA) form to the open source release.
- Added support for `df::thin_film` modifier for glossy BSDFs that feature mode `df::scatter_reflect_transmit` or `df::specular_bsdf` (OM-52209).
- Improved distilling of bsdf color-mixes.
- Added Python bindings wrapper for `IMdl_evaluator_api` to support additional MDL annotations (OM-11581).

- Fixed serialization failures when serializing optimized modules with removed unused functions.

16.3 Fixed Bugs

16.3.1 General

- Issue a fatal error in case the default/fallback-material could not be created (related to nvbugs 3707629).
- Improved contour detection/stability of Toon postprocessing (nvbugs 3724232).

16.3.2 Iray Photoreal & Iray Interactive

- Improved inside-projector-geometry test, which avoids unstable behavior for planes where one coordinate is zero (nvbugs 3621214).

16.3.3 Iray Photoreal

- Fixed artifacts visible in the environment/dome when using the caustic sampler.
- Fixed a potential crash when using guided sampling.
- Fixed missing backplate when switching from AUX to the result buffer.
- Fixed potential errors and crashes when returning too early from `cancel_render`, i.e. while the postprocessing was actually still running. This is now synchronous as intended.
- Improved performance for certain cutout opacity cases when using the caustic sampler (nvbugs 3710624).
- Fixed a discrepancy between evaluation and sampling of thin-walled glossy surfaces.
- Fixed crash for certain multi-/overlapping emissive volume data setups and instancing being turned on.
- Fixed regression on RTX/Turing-and-up cards with fiber rendering (restore originals fiber endcap behavior) (nvbugs 3699759).
- Improved scene error handling (e.g. reject invalid camera transforms before rendering, add a safeguard to reject picking) (related to nvbugs 3692956).
- Fixed disappearing backplate by correctly clamping inverse tonemapped backplate color to zero.
- Fixed a rare issue with render calls being falsely rejected in network rendering mode.
- Optimized linking-phase pre-processing stage if no JIT MDL materials found.
- Improved efficiency of rendering emissive volumes dramatically for complicated setups.
- Fixed missing contributions (i.e. too dark) within uniform emissive volumes.
- Fixed crash when using `df::microfacet_ggx_smith_bsdf`.
- Improved preprocessing speed for multiregion/material objects (nvbugs 3705800).

16.3.4 Iray Interactive

- Fixed a crash when going out of GPU memory (nvbugs 3600421).

16.3.5 Material Definition Language (MDL)

- Fixed loss of tint for specular BSDF (MDL-870).
- Fixed handling of MDL cast operators in the SDK.
- Fixed missing struct constructors of re-exported structure types.
- Fixed handling of the uniform modifier in the MDL module builder that caused correct graph constructions to fail.
- Fixed generation of pattern including thin film.
- Fixed typos and descriptions in support_definitions.mdl.
- Fixed issue with Python bindings and deprecated `IMaterial_definition` interface.

16.3.6 MI importer/exporter

- Added support for custom attributes on textures/volumes to both importer and exporter.
- Avoid exporting (ignored on import) face statements.

17 Iray 2022.0.0 beta, build 359000.876

17.1 Known Issues and Restrictions

- The minimum NVIDIA driver version in order to support the new CUDA and OptiX features is R510 U6 (Windows 512.78, Linux 510.73.05). Note that the R515 driver series unfortunately features a performance regression that only affects Iray 2022.0.0. So when using R515 drivers, only drivers 516.93 (Windows), 515.65.01 (Linux) and up feature the necessary performance fix (nvbugs 3707149 and related to 3678591).
- On Linux and macOS/x86 SSE3 enabled CPUs are now the new minimum requirement.
- Removed support for the already deprecated portal lights in Iray Photoreal. It is now suggested to use the new guided sampling and/or the improved caustic sampler instead.

17.2 Added and Changed Features

17.2.1 General

- Updated general libraries:
 - IndeX Direct:
 - Fixed volume-brick iterator precision issue for very large ray origin positions (leading to the iterator getting stuck).
 - Fixed free sampler node caching (reduced overhead for repeated sampling in close proximity).
 - Severly improved CPU performance.
 - Added tri-linear filtering mode for CPU path.
 - Embree-3.13.2-359470 (with a custom patch for the crashes seen in nvbugs 3643565)
 - OpenVDB 8.0.1
 - LLVM 12.0.1
 - CUDA 11.6
 - OptiX 7.4.1
 - GLEW-2.1.0-358555
 - OpenImageIO (not enabled in the beta yet though)

17.2.2 Iray Photoreal & Iray Interactive

- Added support for the OptiX based upscaling AI denoiser (2x2 upscaling for the moment). If denoising is available and upscaling is requested, the render target canvas in question must be scaled as well. If upscaling does not happen (e.g. because post processing happens on CPU), the original pixels are written to the upscaled location but only fill a quarter of the window.
- Replaced the AI denoiser with the newer OptiX AOV denoiser model for improved image quality.
- Made the resolution requirements for user render target canvases more flexible. Instead of requiring an exact match between the full camera resolution and the resolution of each canvas, each canvas is now only required to fit the offset render window. So canvases may be larger or smaller than the camera resolution as long as the render window fits in its

correct location. In addition, canvases which exactly match the size of the render window but do not leave room for the offset will also be accepted.

- Implemented (optional) recursive picking. This allows picking of all objects along a ray, rather than just the first one.

17.2.3 Iray Photoreal

- Added support for JIT compiled MDL materials in inhomogeneous volumes.
- Added emission support for volumes. This features a fast path as well as JIT support. The fast path supports blackbody emission using a temperature field as specified by the new `base::volume_blackbody_emission` MDL function. Even more efficient emissive volume support (improved sampling) will hopefully be available already with the 2022.0.0 final.
- Implemented a first set of improvements for faster material updates, thus PTX code generation and compilation/linking times are often reduced massively. Note though that updates currently still feature the same main limitation as before, i.e. only full rebuilds are currently supported. This will be addressed in the next releases.

17.2.4 Material Definition Language (MDL)

- Increased the required Python version to 3.8 as this is required by LLVM.
- Added support for building the MDL SDK and the Examples on MacOS on ARM.
- In this version, all backends depend on LLVM now. The LLVM version was lifted from 8.0 to 12.0.1. This change is only visible, if LLVM-IR is chosen as the output format.
- The old GLSL backend was replaced by the new LLVM based one. This new backend supports all capabilities of the HLSL backend, especially:
 - It can compile functions (as the OLD GLSL backend).
 - It can compile materials (by using the libbsdf).
 - GLSL is now at the same level as HLSL, PTX, and native.
- In this version, not all options of the old GLSL backend are supported, in particular the state cannot be configured in the same flexible way as it was possible with the old backend. Only passing the whole state as a struct is currently supported.
- Added support for node attributes in Distiller rules.
- Added support for Distilling to custom target material model.
- Added methods on `IImage_api` to clone canvases and tiles, to convert the pixel type and for gamma correction on tiles.
- Disabling the “materials are functions” feature is now deprecated. To that end, the interface `IMaterial_definition` has been deprecated and most of the interface `IMaterial_instance` has been deprecated. Similarly, for the corresponding enumerators of `Element_type` and `IMdl_evaluator_api::is_material_parameter_enabled()`. See the documentation linked from `IMaterial_instance` for details. The interfaces with the full set of methods and the enumerators are still available if `MI_NEURAYLIB_DEPRECATED_13_0` is defined.
- Added `IMdl_factory::is_valid_mdl_identifier()`. Using this method avoids the need to hardcode the list of MDL keywords yourself. The example “Generate MDL identifier” makes now use of this method.

- Both overloads of `IModule::get_function_overloads()` now also accept the simple name to identify the material or function definition.
- Disabling encoded names is now deprecated. (There are no related API changes, disabling just emits a corresponding warning.)
- The native runtime now blends between frames for animated textures.
- Added methods on `ITriangle_connectivity` and `IAttribute_vector` to transfer data as bulk data.
- Improved `.mi` importer to avoid unnecessary edits when dealing with `material/decal/projector` references.

17.3 Fixed Bugs

17.3.1 General

- Increase maximum supported canvas resolutions (still depends on the chosen pixel format (for now) and if OpenGL is involved in the postprocessing pipeline, but the new maximum is now at 4 GigaPixels).
- Increase maximum supported image output resolution (still limited by the capabilities of freeimage and its used libraries though).
- Register `exclude_from_white_mode` as a known attribute and properly handle it in traverse.
- Fixed missing file path in case of VDB textures returned by `IValue_resource::get_file_path`.
- Fixed remote server shutdown (nvbugs 3645048).

17.3.2 Iray Photoreal & Iray Interactive

- Reduced energy loss for things like `df::custom_curve_layer`, same behavior as already available in libbsdf (OM-27060, OM-39190, OM-44244).
- Denoiser failures, which are not fatal, no longer issue a confusing warning about having run out of memory.
- Improved postprocess de-duplication of canvases. This allows more extensive sharing of passes.
- Fixed alpha channel extraction for mono channel textures (set to 1 now) (MDL-829).

17.3.3 Iray Photoreal

- Replace self intersection handling code, leading to less conservative offsets in some scenes (especially when using instancing), and better self intersection handling/less or no artifacts in others. Please report new regressions or not-yet improved ray tracing precision issues.
- Improved convergence efficiency when both caustic sampler and guided sampling are enabled.
- Improved convergence efficiency when volumes are used in combination with guided sampling.
- Slightly improved memory usage and on average improved convergence efficiency in general when using guided sampling.

- Optimized CPU heterogeneous volume rendering performance.
- Fixed a bug where uninitialized surface volume coefficients could be used.
- Fixed a general SSS/volume precision artifact fix for very dense media (e.g. skin).
- Fixed potential problems as paths start within high density SSS/volumes.
- Made the parameter conversion of ground/global fog consistent (now all parameters respect the scene unit, as before only the density did).
- Fixed invalid zero thickness early-out for thin film (MDL-852).
- Do not intersect objects with primary ray visible flag disabled for the ambient occlusion buffer.
- Avoid some unnecessary geometry updates in flattened geometry mode (instancing off).
- Fixed potentially excessive instancing in auto mode.
- Slightly improve precision of volume rendering.
- Increased verbosity of data handling statistics/log outputs from debug to verbose.
- Improved weight estimation for measured BSDFs. Fixes peaky noise, e.g. for certain AxF-imported carpaints with low albedo measured BSDFs that caused sub-optimal BSDF selection probability.

17.3.4 Material Definition Language (MDL)

- Fixed bug in material hash calculation.

17.3.5 MI importer/exporter

- Ignore more unsupported object/instance/options flags during export.

18 Iray 2021.1.7, build 349500.13092

18.1 Added and Changed Features

18.1.1 General

- Updated general libraries:
 - FreeImage-3.19.x-r1903-OpenEXR-3.1.5-libTIFF-4.4.0-361926 (fixes problems/crashes with huge TIFF metadata TIFFTAG_RICHTIFFIPTC)
- Add versioning to the IndeX Direct libraries.

18.2 Fixed Bugs

18.2.1 General

- Improved contour detection/stability of Toon postprocessing (nvbugs 3724232).

18.2.2 Iray Photoreal

- Fixed artifacts visible in the environment/dome when using the caustic sampler.

18.2.3 Iray Interactive

- Fixed slight precision problems with Oren-Nayar BSDFs (nvbugs 3777916).
- Fixed a crash when going out of GPU memory (nvbugs 3600421).

19 Iray 2021.1.6, build 349500.11420

19.1 Added and Changed Features

19.1.1 General

- Add versioning to the IndeX Direct libraries.
- Updated general libraries:
 - OpenSSL 1.1.1q
 - FreeImage-3.19.x-r1903-OpenEXR-3.1.5-libTIFF-4.4.0-360897
 - FFmpeg-lgpl-4.4.2-360169
 - OpenEXR 3.1.5

19.1.2 Material Definition Language (MDL)

- Added Python bindings wrapper for `IMdl_evaluator_api` to support additional MDL annotations (OM-11581).
- Fixed serialization failures when serializing optimized modules with removed unused functions.

19.2 Fixed Bugs

19.2.1 Iray Photoreal

- Fixed crash when using `df::microfacet_ggx_smith_bsdf`.
- Improved preprocessing speed for multiregion/material objects.

19.2.2 MI importer/exporter

- Avoid exporting (ignored on import) face statements.

20 Iray 2021.1.5, build 349500.10431

20.1 Added and Changed Features

20.1.1 General

- Updated general libraries:
 - Embree-3.13.2-359470 (with a custom patch for the crashes seen in nvbugs 3643565)

20.2 Fixed Bugs

20.2.1 Material Definition Language (MDL)

- Fixed handling of the uniform modifier in the MDL module builder that caused correct graph constructions to fail.
- Fixed generation of pattern including thin film.

21 Iray 2021.1.4, build 349500.10153

21.1 Added and Changed Features

21.1.1 General

- Internal-only release.

22 Iray 2021.1.3, build 349500.9894

22.1 Added and Changed Features

22.1.1 General

- Added a registry `IDebug_configuration` option `index_library_path` which controls where to look for `libnvindex` and its plugins. The default is empty, which leaves the default behavior unchanged.

22.1.2 Material Definition Language (MDL)

- `libbsdf`: Implemented clarified `df::thin_film` specification - it now propagates the thin film coating parameters to its base and in case the base is `df::fresnel_layer` or `df::fresnel_factor`, the correct coating effect on the Fresnel term is computed there (nvbugs OM-33639).

22.2 Fixed Bugs

22.2.1 Iray Photoreal & Iray Interactive

- Clamped light cosines for EDF evaluation due to possible numerical issues (nvbugs 3550059).
- Avoid texture compression on textures that are evaluated using `tex::lookup_float3-reads` (via MDL). This fixes e.g. artifacts on normal maps when using the global texture compression option. The global scene option `compression_override` will now be simply ignored on such textures, but the per texture option for compression still stays intact then (OM-50785).
- Fixed crash in CPU implementation of 8-bit-scalar texel lookup MDL runtime function (MDL-828).

22.2.2 Iray Photoreal

- Fixed a hang when using guided sampling due to an invalid setup of the guidance synchronizer (nvbugs 3602673).
- Fixed initial IOR/volume stack handling for lights and cameras for certain materials (nvbugs 3632721).
- Fixed an issue where caustic sampler paths starting on the environment got wrong IOR/volume stacks assigned (nvbugs 3632718)
- Fixed an inconsistency for paths connecting through transparent surfaces.
- Avoid potential memory leaks when resetting a texture.

22.2.3 Iray Interactive

- Properly override texture compression if `iray_texture_compression` string option is set.

22.2.4 Material Definition Language (MDL)

- Respect scene option texture compression override when determining if MDL resources need an additional alpha channel texture (OM-46894).
- Fixed mipmapping of 2D texture access in the native texture runtime (used with native code generation).
- libbsdf: Fixed numerical corner case in `df::measured_factor` causing broken auxiliary buffer on the native backend.
- Removed double precision computations in libbsdf implementation, causing double type used in HLSL/native/PTX.

22.2.5 MI importer/exporter

- Consider selector when creating image/volume db names.
- Prevent export of standard flags (nvbugs 3607259).
- Gracefully handle disable/visible flags on cameras during import by simply ignoring them (nvbugs 3607259).
- Ignore more unsupported object/instance/options flags during import.

23 Iray 2021.1.2, build 349500.8766

23.1 Added and Changed Features

23.1.1 General

- Updated general libraries:
 - zlib 1.2.12

23.1.2 Iray Photoreal & Iray Interactive

- Animated textures (see Iray 2021.1.0 section below) are now supported in the rendering core, too.

23.1.3 Iray Photoreal

- Added calls to the progress callback during scene preprocessing and access (which includes lazy construction of scene data). This change requires that the application's progress callback survives all rendering. This was always a documented requirement but not enforced or used so far.

23.2 Fixed Bugs

23.2.1 Iray Photoreal & Iray Interactive

- Fixed missing update of section cap colors.
- Avoid accidental evaluation of UDIM normal maps as bump.
- Fixed potentially wrong handling of negative cutout opacity values.
- Fixed access of invalid tag for certain environment materials.

23.2.2 Iray Photoreal

- Fixed striping patterns that appeared when using certain JIT MDL functions.
- Fixed wrong cutout opacity handling for negative or large values in light sampling, leading to wrong emission behavior.

23.2.3 Iray Interactive

- Reduced the automatic normal clamping threshold (which is used to prevent darkening on extreme bump or normal maps), previous setting was too aggressive and gave artifacts at grazing angles. This now matches Iray Photoreal again.
- Avoid restarting rendering if the tonemapper changes while backplate tonemapping is enabled, but with no active backplate (improves on the fix for nvbug 3268527).

23.2.4 Material Definition Language (MDL)

- Remove wrong error message about failures to construct MDL file paths when using the module builder.
- Remove invalid optimization in DAG hashing.

24 Iray 2021.1.1, build 349500.8264

24.1 Added and Changed Features

24.1.1 General

- Updated general libraries:
 - OpenSSL 1.1.1n
 - NVAPI R510
- Support .rgb extension for textures in the SGI file format.

24.2 Fixed Bugs

24.2.1 General

- Fixed recording of pruned journals, which in practice lead to some unnoticed changes in case of overlapping transactions (e.g. missing material updates).
- Added on-demand-meshes to the list of items that are valid for scene updates.
- Fixed processing of integer vector userdata attributes.
- Fixed infinite loop for on demand meshes with a single material.
- Fixed a crash related to removing projectors and its parents from a group.
- Fixed a crash due to a use-after-free issue with static projector arrays.
- Extended error reporting for more CUDA calls (especially on RTX cards).

24.2.2 Iray API

- Fixed handling of memory allocation failures in `IImage_api::create_canvas()/create_tile()` methods.

24.2.3 Iray Photoreal & Iray Interactive

- Fixed a crash when changing texture compression options.
- Fixed infinite loops with the builtin implementation of `base::file_bump_texture` and UDIM input.
- Fixed artifacts in noise band with object space UVW coordinate source.
- Massively improved processing speed of material regions.
- Enforced MDL-JIT compilation in case varying input is found to be attached to uniform parameters.
- Allowed implicit conversion when querying userdata attributes via MDL scenedata lookups rather than requiring the types to match exactly.

24.2.4 Iray Photoreal

- Fixed handling of spectral textures when using GPU rendering (nvbugs 3561235).
- Fixed processing of backplate shaders other than `base::file_texture` (nvbugs 3564192).

- Fixed crash when using the caustic sampler with certain decal combinations (nvbugs 3568867).
- Fixed a rare crash when rendering heterogeneous volumes.
- Improved performance (i.e. less overhead) and memory usage of the new guided sampling once more. It's now even more recommended to enable it, unless the rendered scene(s) are very simple, or one has to rely on deterministic progressive rendering on multi-GPU/hybrid setups (note that guided sampling is unbiased, so only its temporary noise is not fully deterministic).
- Improved heterogeneous volume rendering once more.
- Improved navigation speed by avoiding some unnecessary scene updates.
- Improved spectral volume sampling of uniform volume objects.
- Removed a discrepancy when rendering decals in the contents of auxiliary buffers (between renderings that only render aux buffers and those which also render LPE buffers).
- Optimized LPE buffer processing speed.
- Reduced the automatic normal clamping threshold (which is used to prevent darkening on extreme bump or normal maps), previous setting was too aggressive and gave artifacts at grazing angles.
- Ensured that single frames are completed after a device failure on RTX cards.

24.2.5 Iray Interactive

- Fixed processing of float userdata.

24.2.6 Material Definition Language (MDL)

- Fixed filename extension mismatch when exporting textures referenced from MDL modules. Under certain circumstances, the texture was copied, but got a different filename extension, causing problems importing the MDL module again.
- Fixed creation of function calls of the cast operator if the target type has frequency qualifiers. Similarly, fixed creation of function calls of the ternary operator if the argument types have frequency qualifiers.
- libbsdf: Fixed incorrect child normal orientation usage in `df::color_fresnel_layer`, `df::color_custom_curve_layer` and `df::color_measured_curve_layer` for backside hits.
- libbsdf: provide thin-film Fresnel aware implementations of `df::fresnel_factor`, `df::fresnel_layer`, and `df::color_fresnel_layer`.
- Also encode the simple name of function definitions. For almost all functions this does not make any change since the simple name is usually an identifier, except for a couple of operators from the builtins module.
- HLSL backend: Fixed code generation for scene data access functions inside automatically derived expressions.

24.2.7 MI importer/exporter

- Further optimized the import/parsing of .mi files.

- Optimized the processing of scenes with a lot of material references (e.g. if few/the same material(s) are specified per triangle).

25 Iray 2021.1.0, build 349500.7063

25.1 Known Issues and Restrictions

- There is a known issue with older drivers (496.XX) and Iray Interactive (potentially Photoreal, too, although not reported so far), leading to crashes on some setup/scene combinations (nvbugs 3418345). Newer driver branches (510.XX and up) fix this regression again.
- Iray Interactive will now also no longer internally duplicate materials (similar to Photoreal) that feature a certain dependence on MDL state transforms: Most real-world materials depend on state transforms, but the case where state transforms are used inside an uniform MDL expression is extremely rare (if used at all). So most of the time, materials were duplicated without the need for it, resulting in a significant performance loss during pre-processing/material conversion, especially for large scenes. This is a performance workaround for now and will hopefully be fixed properly later-on with improved compiler support.

25.2 Added and Changed Features

25.2.1 General

- Updated general libraries:
 - SQLite 3.37.0
 - USD 20.08 (for the optional USD exporter)
- Write Bridge version to the Windows FileProperties Dialog: It will appear after the Product Version as XXXXXX.YYYY (Bridge Version: XXXXXX.ZZZ).
- Disabled WEBP export in the FreeImage plugin due to memory access violations.

25.2.2 Iray API

- Added support for animated textures.
 - The signature of various methods on IImage and ITexture has been changed. The frame index has been added as first parameter and the order of uvtilde index and mipmap level has been flipped. The default arguments have been removed. The old signatures are still available if MI_NEURAYLIB_DEPRECATED_12_1 is defined. Methods to query the mapping between frame index and frame number have been added.
 - The method uvtilde_marker_to_string() on IMdl_impexp_api and IExport_api has been renamed to frame_uvtilde_marker_to_string(). It is still available under the old name if MI_NEURAYLIB_DEPRECATED_12_1 is defined. The method uvtilde_string_to_marker() on both interfaces has been deprecated without replacement. The last component of IImage::get_original_filename() is an alternative (if available), or construct a custom string from scratch.
 - The interface IMdl_resolved_resource has been split such that it represents an array of instances of the new interface IMdl_resolved_resource_element, where each element corresponds to a texture frame (only one for non-animated textures and other resources).

- The frame parameter has been added to various method of the runtime interfaces `Texture_handler_vtable` and `Texture_handler_deriv_vtable`. A new member `m_tex_frame` to support the intrinsics `tex::first_frame()` and `tex::last_frame()` has been added.
- Added an overload of `IMdl_factory::clone()` that allows cloning of an execution context. Useful for local changes to the context options.
- Changed `mi::math::gamma_correction()` to include the alpha channel, as it is done already in other places.
- Added methods on `ITriangle_mesh` to transfer points and triangle indices as bulk data.

25.2.3 Iray Photoreal & Iray Interactive

- The implementation of the `df::thin_film` has been improved. It now acts as a modifier to account for thin-film-coating of a Fresnel effect in its underlying base (affected bases are `df::fresnel_layer` and `df::fresnel_factor`, i.e. it is possible to coat dielectrics and metals).

25.2.4 Material Definition Language (MDL)

- MDL 1.7 Language Specification
 - Removed the draft status of this document.
 - Clarified that constants can be declared locally as well, which is working since MDL 1.0.
 - Clarified that the `thin_film` modifier applies only to directly connected BSDFs that have a Fresnel term.
 - Clarified that the `state::normal()` orientation is facing outward from the volume, with thin-walled materials treated as enclosing an infinitesimally thin volume, such that both sides are facing outward and the normal always points to the observer.
- The MDL core compiler can resolve frame sequences using frame markers in resources.
- The JIT backend issues an error message if the user specifies a state module that does not contain all necessary functions.
- The MDL core compiler avoids a redundant call to the entity resolver when importing modules.

25.2.5 MI importer/exporter

- Improved `.mi` importer speed for binary geometry data. Roughly twice as fast now.
- Enabled binary export in the `.mi` exporter by default for meshes with ≥ 1000 triangles. Configurable with the exporter option `"mi_write_binary_vectors_limit"` of type `UInt32`.

25.3 Fixed Bugs

25.3.1 General

- Fixed bloom radius in presence of windowed rendering.

- Fixed incorrect error estimate when rendering camera windows in combination with vignetting through a tonemapper.
- Fixed missing leaf material update when scenedata names change (e.g. causing Iray Interactive to not react on scenedata name changes).
- Improved texture compression for low-variation areas, including "wobbly" behavior of some normal maps.
- Optimized atomics/atomic usage, especially on Arm based platforms.
- SIMD optimized Arm based CPU image conversion and postprocessing.
- Fixed gamma value of pink 1x1 default textures.

25.3.2 Iray Photoreal & Iray Interactive

- SIMD optimized Arm based CPU rendering.
- Fixed numerical issues using Arm based CPU rendering (e.g. specular surfaces turning black).
- Fixed tangent_u orientation in case the normal is flipped for MDL state on backside.
- Adapted surface normal flipping to the clarified MDL spec: Flip normals to front side if non-thin-walled, flip to ray for thin-walled.
- Fixed numerical issue for `df : : fresnel_factor (ior = 0)`.
- Fixed wrong access of environment, backplate, aperture and projector functions that have been invalidated due to an MDL module reload (nvbugs 3459568).
- Remove unnecessary warnings on platforms without CUDA support (i.e. macOS).

25.3.3 Iray Photoreal

- Further optimize convergence speed and robustness of the new guided sampling option. It's now even more recommended to enable this option (at least) for interior scenes.
- Parallelize parts of the geometry import code.
- Fixed temporary artifacts on RTX GPUs when navigating interactively at small resolutions and high sample rates (nvbugs 3418824).
- Fixed caustic sampler paths being clipped by the camera near plane.
- Fixed regression that lead to unnecessary rendering restarts on some parameter changes (e.g. en/disabling the AI denoiser).
- Fixed artifacts with enabled caustic sampler + rendering irradiance buffers for specular surfaces.
- Fixed caustic sampler + rendering irradiance buffers when connecting through other surfaces.
- Fixed empty (zero vertices) fiber objects on RTX GPUs.
- Fixed renderer scenedata from being accidentally processed when used with wrong type.
- Fixed missing check if tag is valid and actually corresponds to a function call, leading to crashes with volumes (nvbugs 3491070).

- Fixed a tangent orientation flip in the MDL state update on a `state::normal()` change when using the JIT material path, e.g. incorrect JIT layer bumps in case there is a main geometry bump (Jira MDL-788).
- Optimize inhomogeneous volume rendering performance.
- Slightly improved performance of some GPU updates/uploads.
- Slightly improved convergence when rendering volumes/SSS.
- Slightly optimized detection of section object/material changes.
- Slightly optimized geometry preprocessing.

25.3.4 Iray Interactive

- Properly support the optional handling of energy loss for diffuse BSDFs with applied bump/normal mapping. This improved behavior can be deactivated by the scene option `iray_diffuse_bump_energy_compensation` (nvbugs 3459865).
- Fixed issue if number of scenedata names used in MDL-JITted functions has changed.
- Greatly reduced time spent in texture loading for scenes with a large number of objects (nvbugs 3459561: 1 hour down to 13 seconds!).
- Due to the missing duplication of some materials (see Restrictions section above), some scenes with overlapping volumes/SSS-materials will now render more correctly (a sideeffect of the simple IOR stack model of Interactive).

25.3.5 Material Definition Language (MDL)

- Fixed incorrect normal flip for strongly bumped normal input (libbsdf, OM-37821).
- libbsdf: Fixed incorrect flipping of the shading normal for strongly bumped normals. Note that libbsdf requires that state input shading and geometric agree on sidedness (it has been forgiving with respect to that due to this bug).
- libbsdf: Fixed a numerical issue for `df::fresnel_factor()` (for `ior == 0`).
- libbsdf: Fixed implementation of albedo for `df::tint(reflect, transmit)`.
- Fixed a race condition for accessing global core JIT backend options from different threads, which could have caused overwritten options or a crash, by using the thread local core thread context options instead.
- Fixed handling of resource sets if used inside MDLE archives.
- Fixed a crash inside the MDL core compiler if a material preset with too many arguments is compiled.

26 Iray 2021.1.0 beta, build 349500.5279

26.1 Known Issues and Restrictions

- Deprecated support for Light Portals. These will be removed in the next major version release. Instead, the new automatic guided sampling option should be employed wherever feasible.
- Removed support for the special/limited Nitro rendering mode.

26.2 Added and Changed Features

26.2.1 General

- Native macOS Arm support now available (still considered experimental for this beta phase, but should be fully functional, so please test). In addition, all macOS builds are now completely clean of any CUDA dependencies/libraries.
- Updated general libraries:
 - FreeImage-3.19.x-r1859-openexr-2.5.3-libtiff-4.1.0-354007 (fixes crash with certain TIFF files, nvbugs 200765028, and has LibRawLite removed)
 - Boost 1.77
 - FFmpeg-lgpl-4.4.1-353186
 - NVAPI R495
 - Embree 3.13.2
 - GLEW-2.1.0-348116
- Enabled better compression levels for the PNG/BMP/TIFF/TGA formats in the image plugin.
- New (but for now experimental/available on demand) USD exporter plugin.

26.2.2 Iray API

- Added support for texture selectors. The selector can be queried with methods on `IImage`, `ITexture`, `IVolume_data`, `IValue_texture`, and `ITarget_code`. For non-volume data textures, the supported selectors are restricted to "R", "G", "B", and "A" for now.
- Renamed `IVolume_data::get_channel_name()` to `IVolume_data::get_selector()` for consistency with `IImage` and `ITexture`. The old method is still available if `MI_NEURAYLIB_DEPRECATED_12_1` is defined.
- The signature of `IMdl_factory::create_texture()` has been extended to specify the selector. The old signature is deprecated and still available if `MI_NEURAYLIB_DEPRECATED_12_1` is defined.
- Added `IImage_api::create_canvas_from_reader()` to enable canvas creation directly from a reader (in addition to `create_canvas_from_buffer()`).
- Added the methods `get_pixel_type_for_channel()` and `extract_channel()` to `IImage_api`, which are useful for extracting RGBA channels from existing textures.
- Improved resource enumeration on modules. The new method `IModule::get_resource()` returns an `IValue_resource` with all details, including gamma mode and

selector. The old methods returning the individual bits are deprecated and still available if `MI_NEURAYLIB_DEPRECATED_12_1` is defined.

- Added overloads of `IValue_factory::create()` that accept a range annotation as argument, and a type and an entire annotation block. This makes it simpler to create values that observe range annotations. Modified `Definition_wrapper` to make use of the latter if a parameter has no default.
- Added `IFunction_call::reset_argument()` which sets an argument back to the parameter default (if present), or an value observing given range annotations, or a default-constructed value.
- Extended `IValue_factory::compare()` and `IExpression_factory::compare()` to support floating point comparisons with an optional epsilon range.
- Improved documentation about database limitations, see “Database access” in the API reference manual.
- The materials-are-functions feature is now enabled by default. See the documentation referenced from `IMdl_configuration::set_materials_are_functions()`. All examples have been updated to avoid usage of `IMaterial_definition` completely, and `IMaterial_instance` as much as possible.
- Added an `user_data` option to the `IMdl_execution_context`, which allows the user to pass its own interface pointer around, in particular to the methods of `IMdl_entity_resolver`.
- Improved performance of editing instances of `IMdl_function_call`, in particular for instances with a large set of arguments. Depending on the exact circumstances, this can cut the time for a full edit cycle (create transaction, create argument editor, change argument, destroy argument editor, commit transaction) in half. An additional speedup can be obtained by making use of the additional optional argument of `Argument_editor`.
- Extended `IExpression_factory::compare()` to support deep comparisons of call expressions. Useful to figure out whether an exporter needs to export an argument, or can rely on the corresponding default on the definition.
- Export to EXR takes now the quality parameter into account: a value of 50 or less selects half as channel type.
- Added `create_reader()` and `create_writer()` to `IMdl_impexp_api`.
- Changed the behavior of `IImage_api::adjust_gamma()` to include the alpha channel. This also affects export operations (if `force_default_gamma` is set) and the MDL texture runtime if derivatives are enabled.

26.2.3 Iray Photoreal & Iray Interactive

- Extend photographic tonemapper:
 - New `mip_burn_highlights_blended_component` mode to linearly blend between the behavior of `mip_burn_highlights_per_component` and `mip_burn_highlight_max_component`. Using this mode, one can control by how much highlights de-saturate (via `mip_burn_highlights_saturation`).
 - Employ a more flexible compression curve, controlled via `mip_compression_variant`. Default is `reinhard` and yields the previous behavior, controlled by

`mip_burn_highlights`, whereas with `raw_parameters` one may manually specify coefficients for the internal curve via `mip_compression_parameters0` and `mip_compression_parameters1`. In addition, via `uncharted2` the popular game engine tonemapper with default parameters is constructed, and via `ue4_aces` the ACES variant used in Unreal Engine 4 is used.

- Optional handling of energy loss for diffuse BSDFs with applied bump/normal mapping. This improved behavior can be deactivated by the scene option `iray_diffuse_bump_energy_compensation` with
 - `none`: no compensation
 - `brdf`: only diffuse reflection is compensated
 - `btdf`: only diffuse transmission is compensated
 - `all`: compensate both (default)

26.2.4 Iray Photoreal

- Added new guided sampling scheme via `guided_sampling`. Enabling this option will usually help convergence speed of interior or other complicated scenes at the price of slightly increased rendering times per iteration and higher memory usage. Thus, simple scenes such as turntable-like objects will not profit, and may even render a bit slower. The exact improvements also depend a lot on the used hardware (e.g. multi GPU or hybrid CPU/GPU rendering may not profit as much as single/dual GPU setups) and if batch or interactive scheduling is used. Also note that disabling the firefly filter may sometimes lead to more high frequency noise such as speckles. More quality and performance improvements to this new sampling scheme will be implemented for the 2021.1.0 final release.
- Added support for texturing volume coefficients on the surface via scene option `iray_allow_surface_volume_coefficients`.
- Added new exponential inhomogeneous ground/global fog support via `iray_ground_fog` (see documentation for additional configuration parameters).
- Added scene option attribute for custom photometric spectral observer: see `iray_spectral_observer "custom"` and `iray_spectral_observer_custom_curve <values>` in the documentation.
- Added additional fisheye based camera distortion model (incl. Depth of Field support) via `mip_lens_distortion_type "equidistant"`.
- Added support to control section cap colors on a per clip plane basis.
- Added spectral blackbody support.
- Added support for the MDL hair BSDF also on non-fiber geometry. Note that fiber geometry in all cases is still the preferred solution to use the hair BSDF. Triangulated geometry should only be used as a last resort (e.g. on legacy scenes/assets).
- De-duplicate materials in order to speed up material import for scenes with a large number of duplicate material instances (e.g. as currently generated via some Omniverse scenes).
- Added more fine-grained control over matte shadow opacity: Replaced global `matte_shadow_affects_alpha` scene option as the means to control alpha opacity of matte shadows by an LPE-based approach.

Scenes which set `matte_shadow_affects_alpha` to false will now trigger a warning and the alpha channel will contain matte shadows. Such scenes are returned to the original result by setting `iray_default_alpha_lpe = "[LeLmLms]T*E"`, provided that they do not already use any specialized alpha LPEs. In that case, adding Lms to the list of captured light types will generally remove matte shadows again.

If invoked as before, the utility functions for alpha LPEs provided by `IRendering_configuration` now yield expressions which do not have opaque matte shadows. Matte shadows can be made opaque by passing `ALPHA_TRANSMIT | ALPHA_MATTE_SHADOW` (or similar) to these functions.

26.2.5 Iray Interactive

- Added support to query and respect the new 2021.0.0 `iray_texture_compression` scene option (nvbugs 200777016).

26.2.6 Material Definition Language (MDL)

- The module `::core_definitions` requires now MDL 1.6 and contains a new material `surface_falloff`.
- Added a warning to catch some wrong implementations of `IMdl_resolved_module::get_module_name()`.
- Added support for libbsdf normal adaption for CUDA and native runtime.
- The libbsdf implementations of the functions `df::diffuse_reflection_bsdf` and `df::diffuse_transmission_bsdf` now compensate for energy loss caused by differences of shading and geometric normal (OM-37821).
- Removed all internal MDL distiller targets except for the "none" target.

26.3 Fixed Bugs

26.3.1 General

- Fixed `network_configuration` shutdown deadlock.
- Fixed missing journal flags for volumes.
- Increase maximum web socket payload size to 2GB. In the future there will be a proper API call to steer the maximum payload size (nvbugs 3447753).

26.3.2 Iray API

- Fixed `IFactory::clone()` when dealing with untyped arrays (nvbugs 3383204).
- Fixed `IMdl_backend::deserialize_target_code()` such that the internal DF texture are no longer missing under certain circumstances.

26.3.3 Iray Photoreal & Iray Interactive

- Optimized blackbody emission.
- Further fixed thin film factor implementation.
- Added support for SSE/AVX accelerated tone mapping (if no NVIDIA GPUs are available).

26.3.4 Iray Photoreal

- Further fixed shadow terminator related artifacts, like tessellation of geometry becoming visible or other lines or marks on curved surfaces (nvbugs 200753581).
- Fixed a crash when trying to load a non-existing field from a OpenVDB file.
- Fixed motion blur in presence of disabled section planes.
- Fixed hangs in inhomogeneous volume rendering.
- Fixed interaction of section planes and cutouts: Planes with active `clip_light` did not work after cutouts (nvbugs 200774869).
- Fixed wrong shadows of section plane caps.
- Fixed section plane caps of clipped objects with enabled cutouts (e.g. also leading to wrong shadows).
- Fixed cleanup of instanced triangle data, leading to potential waste of memory.
- Fixed too bright matte object paths for very short path length restrictions.
- Fixed an issue with the tonemapper producing NaNs.
- Ignore volume first-hit visibility, as the visible flag does not relate well to volumes.
- Improved robustness of inhomogeneous volume rendering.
- Improved rendering performance for certain volume combinations and dense volume data.
- Improved material updates if the `iray_texture_compression` option changed.
- Report OptiX messages from a dedicated module, rather than IRAY or POST, for greater clarity.

26.3.5 Iray Interactive

- Further fixed some potential data races and multi-threading issues which could cause use-after-free errors in the garbage collection (nvbugs 3379659 and 200744281).
- Fixed update of host textures. Not all texture parameters were copied on updates from the shared texture element. In consequence, if device assignment changed, the CPU "inherited" one parameter from the GPU texture previously in some cases (nvbug 3386411).

26.3.6 Material Definition Language (MDL)

- Fixed support for JIT-compiled functions for coordinate projections with transforms.
- Improved numerical stability in `base::coordinate_projection()`.
- Improved performance of `math::blackbody()` implementation.
- Fixed incorrect BSDF evaluation for `df::sheen_bsdf` with a transmitting `multiscatter` BSDF parameter (OM-32211).
- Further fixed thin film factor implementation.
- Fixed handling of weak imports for MDL < 1.6 if an external entity resolver is set. The semantic of weak imports is now handled by the core compiler itself, an external entity resolver sees now only absolute or strictly relative module names.
- Fixed distiller crash with certain `mrule` transformation rules.

- Fixed distiller crash with certain combinations of distribution functions in distilled materials, e.g., `df::directional_factor` in material emissions.
- Fixed `ICompiled_material::get_hash()` to take the material slots for `surface.emission.mode` and `backface.emission.mode` into account. Added enumerators `SLOT_SURFACE_EMISSION_MODE` and `SLOT_BACKFACE_EMISSION_MODE` to `mi::neuraylib::Material_slot`. Also added `SLOT_FIRST` and `SLOT_LAST` to support easier enumeration over all material slots.
- Fixed a crash with default constructed BSDF measurements during MDL export/MDLE creation.
- Removed `const` modifier on distilled materials such that they can be stored in the database.

26.3.7 MI importer/exporter

- Fixed escaping mechanism: Escaping characters (like quotes) now also works for the escape character.
- Fixed support for standard flags of volume objects, e.g. hidden volumes.

27 Iray 2021.0.5, build 344800.12856

27.1 Fixed Bugs

27.1.1 General

- Fixed bloom radius in presence of windowed rendering.
- Improved texture compression for low-variation areas, including "wobbly" behavior of some normal maps.

27.1.2 Iray Photoreal & Iray Interactive

- Fixed artifacts in noise band with object space UVW coordinate source.

27.1.3 Iray Photoreal

- Fixed caustic sampler paths being clipped by the camera near plane.
- Fixed crash when using the caustic sampler with certain decal combinations (nvbugs 3568867).
- Fixed temporary artifacts on RTX GPUs when navigating interactively at small resolutions and high sample rates (nvbugs 3418824).
- Fixed empty (zero vertices) fiber objects on RTX GPUs.

28 Iray 2021.0.4, build 344800.9767

28.1 Fixed Bugs

28.1.1 General

- Increase web socket payload size by 10x. In the future there will be a proper API call to steer the maximum payload size.

28.1.2 Iray Photoreal & Iray Interactive

- Fixed a potential endless loop for complicated materials (nvbugs 3430252).

28.1.3 Iray Photoreal

- Fixed an issue with too transparent alpha channel values with SSS materials.
- Apply thin film modifier on both sides, now consistent with Iray Interactive again (nvbugs 3444673).
- Fixed spectral IOR component selection in thin film computation.

28.1.4 Material Definition Language (MDL)

- Fixed a rare case of incorrect handling of user-defined type names for structs and enums when encoded names were enabled.
- Fixed non-deterministic behavior with sincos calls.

29 Iray 2021.0.3, build 344800.8726

29.1 Known Issues and Restrictions

- Iray Photoreal will no longer internally duplicate materials that feature a certain dependence on MDL state transforms: Most real-world materials depend on state transforms, but the case where state transforms are used inside an uniform MDL expression is extremely rare (if used at all). So most of the time, materials were duplicated without the need for it, resulting in a significant performance loss during pre-processing/material conversion, especially for large scenes. This is a performance workaround for now and will hopefully be fixed properly later-on with improved compiler support.

29.2 Added and Changed Features

29.2.1 General

- Updated general libraries:
 - FFmpeg-lgpl-4.4.1-353186

29.3 Fixed Bugs

29.3.1 Iray Photoreal & Iray Interactive

- Fixed crashes if more than one million texture structs were created.
- Fixed inversion of crush blacks parameter in the tonemapper that lead to saturated blue/red pixels.

29.3.2 Iray Interactive

- Improved consistency of bump maps on front- and backside, as bump mapping behavior was inconsistent (e.g. both sides convex or both sides concave, instead of one convex and one concave), similar as Iray Photoreal in 2021.0.2.
- Fixed the corner case of area lights with zero area, leading to infinite emission and even crashes(nvbug 3410220).

29.3.3 Material Definition Language (MDL)

- Apply thin film only if thickness > 0.0 (libbsdf).
- Fixed a bug in the distiller which could lead to a crash for certain materials.

30 Iray 2021.0.2, build 344800.7839

30.1 Added and Changed Features

30.1.1 General

- Updated general libraries:
 - OpenSSL 1.1.1l
 - FreeImage-3.19.x-r1859-openexr-2.5.3-libtiff-4.1.0-350282 (fixes crash with certain TIFF files, nvbugs 200765028)

30.1.2 Iray Photoreal & Iray Interactive

- Added scene option to control the color of the "white mode" global diffuse material `white_mode_color`.
- Added scene option to control the color of the "white mode" global diffuse material in the AOV BSDF weight buffer `white_mode_bsdf_weight`.

30.1.3 Iray Photoreal

- Added motion blur support for inhomogeneous volumes.

30.1.4 Iray Interactive

- Added support for the "white mode" global diffuse material.

30.1.5 MI importer/exporter

- Added im/export of `vector[]` attributes.

30.2 Fixed Bugs

30.2.1 General

- Avoid infinite recursion in some misconfiguration cases of canvas annotations (e.g. setting `iray_default_alpha_lpe` to an empty string, nvbugs 200764440).
- Creation of compiled materials fails now (instead of crashing) if some internal error condition is detected (OM-37446).

30.2.2 Iray Photoreal & Iray Interactive

- Further fixed thin film factor implementation.
- Optimized certain materials using multiple bump maps.

30.2.3 Iray Photoreal

- Fixed caustic sampler ignoring box-typed environment domes.
- Fixed shadow terminator related artifacts, like tessellation of geometry becoming visible or other lines or marks on curved surfaces (partially fixes nvbugs 200753581).
- Fixed interaction of ground plane, scene bounding box, and section plane caps leading to wrong shadows (nvbugs 3342399).

- Fixed small energy loss for `df::sheen_bsdf` in case the multiscatter parameter is the default `df::diffuse_reflection_bsdf`.
- Fixed issue with blends if input texture is spectral by making the builtin color math function spectrally-aware (nvbugs 200763772).
- Fixed partially ignored LPE for matte fog.
- Fixed a change in matte object shadowing in combination with the ground plane (nvbugs 200727716).
- Fixed hangs in inhomogeneous volume rendering.
- Improved consistency of bump maps on front- and backside, as bump mapping behavior was inconsistent (e.g. both sides convex or both sides concave, instead of one convex and one concave).
- Improved precision of result/buffer merges, e.g. for extremely huge sample/iteration counts (nvbugs 200761085).
- Improved performance for object flags preprocessing.
- Improved rendering performance for some JIT-compiled MDL materials.
- Improved efficiency of the caustic sampler a bit.

30.2.4 Iray Interactive

- Fixed missing low-frequency contribution of the Sun & Sky environment to indirect illumination, leading to darker images (nvbugs 3263562).
- Fixed issues with section capping with enabled ground plane (nvbugs 3357191).
- Fixed handling of section planes and section capping for glossy ground reflections (nvbugs 3357191).
- Fixed some potential data races and multi-threading issues which could cause use-after-free errors in the garbage collection (nvbugs 200744281).

30.2.5 Material Definition Language (MDL)

- Fixed incorrect BSDF evaluation for `df::sheen_bsdf` with a transmitting "multiscatter" BSDF parameter (OM-32211).
- Fixed `df::thin_film` implementation for the case material IOR < thin film IOR (libbsdf).
- Changed an internal compiler crash to proper error reporting.

30.2.6 MI importer/exporter

- Fixed line sizes \geq 64K characters (nvbugs 200763394).
- Exporter: Only omit global namespace prefix if we are in global scope (nvbugs 200769503).

31 Iray 2021.0.1, build 344800.4174

31.1 Added and Changed Features

31.1.1 General

- Reduced memory usage for DDS textures with subformat BC7.
- Updated general libraries:
 - CUDA 11.4.1 (which fixes a crash/ endless loop seen (at least) on T4 GPUs (nvbugs 200744281))
 - OpenVDB 8.0.1
- Remove dependency on NVRTC libraries and so being able to remove them again from the release.

31.1.2 Iray Photoreal & Iray Interactive

- Vastly improve texture compression: Much better quality, especially on the high setting.
- Texture compression will now also take advantage of available GPUs for much faster compression/pre-processing.
- Fixed/Replaced thin film factor implementation, as the previous implementation had some downsides:
 - No phase shift from reflection at higher IOR.
 - Intensity was off.
 - Only a single bounce was considered.

31.1.3 Iray Photoreal

- Added ability to exclude scene elements from picking via the new flag `picking_disabled` which can exclude objects, lights, and volumes from picking.

31.1.4 Material Definition Language (MDL)

- Reduced compilation times (Jira OM-33327).
- Updated `nvidia::core_definitions` with new functionality.
- Fixed thin film factor implementation (libbsdf, Jira OM-33639).
- Added a new execution context option “warning” to silence compiler warnings or to promote them to errors. See `IMdl_execution_context` for details.
- Disabled warnings for deprecated MDL materials and functions by default.

31.2 Fixed Bugs

31.2.1 General

- Disabled copying of existing OpenGL image canvas contents to fix camera window copying. Note that this limits some use cases of rendering camera windows for the OpenGL case, as the content outside the window can not be kept around (nvbugs 3319637).

- Skip on-demand meshes during displacement updates as these currently do not support displacement.
- Catch nullptr scene string option.

31.2.2 Iray Photoreal & Iray Interactive

- Don't miss the tint of `df::sheen_bsdf`'s base in case this is a diffuse BRDF.

31.2.3 Iray Photoreal

- Fixed crashes/regression when moving objects in auto instancing mode (nvbugs 200754744).
- Fixed crashes on flag updates in scenes featuring volume data.
- Fixed potential deadlock in texture loader.
- Fixed hangs in inhomogeneous volume rendering (nvbugs 200750968). Also improves zero skipping.
- Don't be able to pick disabled volumes.
- Fixed caustic sampler camera connections ignoring inhomogeneous volumes in almost all cases.
- Fixed caustic sampler shading normal correction to be view dependent (leading to unwanted faceting).
- Fixed caustic sampler connections through thin-walled surfaces being too bright.
- Fixed a crash of the caustic sampler in rare circumstances (floating point precision issue).
- Allow backplate meshes to disable picking.
- Decals now also respect the white mode flag.
- Fixed a rare potential crash when using thin-film/dispersion.
- Free unused scene data on a GPU device, if the device fails. Data was previously marked as unused but garbage collection was not run until the next scene change.
- Fixed another item count discrepancy caused by directional lights not being considered as objects (nvbugs 3343419).
- Demote active SLI message from error to warning.
- Fixed matte shadow intensity array size (nvbugs 3343419).
- Improved general geometry pre-processing performance.
- Minor optimization of fiber pre-processing.

31.2.4 Iray Interactive

- Fixed texture issues when rendering a scene with (unsupported in Interactive) VDB volumes.

31.2.5 Material Definition Language (MDL)

- Fixed handling of `nvidia::baking` annotations.
- Fixed lambda results handling in single-init mode for non-HLSL.

- Fixed uncomputed cosine in sheen_bsdf's multiscatter (was broken for a transmitting multiscatter component, libbsdf).
- Fixed missing `color_weighted_layer()` inside the transmission analysis (bug 19630).

31.2.6 AIX importer

- Fixed crash when loading files with more than 127 data frames.

31.2.7 MI importer/exporter

- Fixed reading of NaN and Infinity values (nvbugs 200744402).
- Fixed a rare crash in the .mi importer.
- Fixed a crash when the name of a scene element clashes with the name of a default of an MDL material or function definition (nvbugs 3342399/3342420/200757058).
- Allow group flags to appear both before and after the item list (nvbugs 200759499).
- Initialize Index Direct when starting .mib importer (nvbugs 3342441).
- Export on-demand-meshes as triangle meshes (rather than just ignoring them).

31.2.8 Deep Learning based Denoiser

- Fixed a potential issue when denoising large resolutions (8K and up).

32 Iray 2021.0.0, build 344800.2052

32.1 Added and Changed Features

32.1.1 General

- Switched to static CUDA runtime, so no need to ship the CUDA runtime libs (libcudart.so/.dll/.dylib) anymore. Note that macOS still needs the dynamic libraries (for now).
- Updated general libraries:
 - CUDA 11.3
 - NVAPI R465
 - JsonCpp 1.9.4 (this one was already used in older releases, even including 2020.0, but the version number was still at 1.9.3)
 - OpenVDB 7.2.2
- Added new scene-option `iray_texture_compression` with values "off", "medium" or "high": This globally overrides texture compression options as given on the `ITexture` itself.
- Added support for channel selectors for VDB-volumes defined in the `texture_3d` constructor inside MDL files.
- Build `openvdb_integration` plugin on all supported platforms.

32.1.2 Iray API

- Extended `ISimple_mesh` interface to support userdata arrays.

32.1.3 Iray Photoreal

- Added previously missing CPU support of IndeX Direct for inhomogeneous volume rendering.
- Added volume picking.
- Added release of host texture-tile memory right after the internal host texture conversion, except for environment textures and backplates to avoid duplicated host texture memory usage. Note that this only works for textures that support lazy loading so that the data can be re-fetched from disk again if needed.
- Added auxiliary ambient occlusion buffer support. Its parameters are controlled by (a subset of) existing Iray Interactive's scene options:
 - `irt_ambient_falloff`
 - `irt_ambient_falloff_min_distance`
 - `irt_ambient_falloff_max_distance`

But now also aliased from new options without the `irt_` prefix.

32.1.4 Material Definition Language (MDL)

32.2 Added and Changed Features

- MDL 1.7 Language Specification

- OpenVDB has been added as supported 3D texture format.
- Minimum required versions have been added for OpenEXR, Ptex, and OpenVDB to conform to the VFX Reference Platform CY2021.
- Supported texture selector string values have been added for each supported texture format.
- Texture sequences with a sequence marker have been added to the definition of texture file paths.
- The auto placeholder type specifier has been added for variable declarations and function return types.
- The float type is required to use a 32-bit representation following the IEEE 754 single precision standard. Floating point operations for float and double may deviate but shall not interrupt nor terminate processing.
- The int type is required to be a 32-bit signed integer in two's complement form with wrap-around operations and without exceptions.
- The restrictions on the texture types, light profile data type, and measured BSDF data type have been removed. They can now be used for local variable types, return types, field types in user defined structures, and element type of arrays.
- A selector string parameter has been added to the texture_2d and texture_3d constructors to support texture formats with multiple data sets and channels in a file. The anno : :usage standard annotation can be used on texture parameters to pre-select selector values in an integration.
- The operators =, ==, and != have been added for the texture types, light profile data type, and measured BSDF data type.
- Emission has been added to the volumetric material properties.
- The return type of a material definition can have an annotation.
- The description of various standard annotations, like in_group and ui_order, mention their wider applicability to more elements in MDL.
- The usage standard annotation on materials is recommended to be used on the return type and not the material definition to match the recommendation for function declarations.
- The hyperbolic trigonometric functions cosh, sinh, and tanh have been added to the standard math library.
- The re-interpreting bit-cast functions float_bits_to_int and int_bits_to_float have been added to the standard math library.
- The offset functions width_offset, height_offset, and depth_offset have been added to give full access to OpenVDB bounding box information.
- The functions first_frame and last_frame have been added to give access to the texture animation bounds.
- The transform function grid_to_object_space has been added to give access to OpenVDB bounding box information in MDL object space.
- A frame parameter has been added to the width, height, depth, width_offset, height_offset, depth_offset, and grid_to_object_space texture functions to select frames in texture sequences.

- A frame parameter has been added to the `texture_2d` and `texture_3d` variants of the `lookup_ltype` and `texel_ltype` family of texture function overloads to select frames in texture sequences.
- The uniform modifier has been removed from the `tint` parameter of the EDF tint modifier.
- The VDF tint modifier has been added.
- An overloaded variant of the `directional_factor` modifier has been added for EDFs.
- The `sheen_bsdf` has been changed to a modifier by adding a `BSDF_multiscatter` parameter.
- The uniform modifier has been removed from the `weight` field of the `edf_component` and `color_edf_component` structures and the upper limit has been removed for the `weight`.
- The `color_vdf_component` structure and VDF overloads for the `color_normalized_mix` and `color_clamped_mix` mixing distribution functions have been added.
- The mix distribution functions `unbounded_mix` and `color_unbounded_mix` have been added for BSDF, EDF, and VDF.
- An Appendix F has been added defining MDL search path conventions.
- A Python binding for the MDL SDK has been added. This binding consists of a shared library generated from the API headers using SWIG, and some additional helper functions for ease of use. In addition, there is a stub Python module to make that shared library accessible from Python code. See "Language Bindings" in the API reference for more information. Examples to demonstrate working with the Python binding are included as well.
- In the API, the array constructor now uses –as all other function definitions– named arguments instead of positional arguments. The previously irrelevant parameter names are now enforced to be "0", "1", and so on.
- The new naming scheme for MDL entities in the `.mi` file format has been enabled by default when encoded names are enabled (see `IMdl_configuration::set_encoded_names_enabled()`). The exporter option `mi_mdl_new_naming_scheme` can be used to control that behavior.
- Added support for the "target model" compilation mode.
- Added a context option `"remove_dead_parameters"` to control the removal of dead parameter in instances of `ICompiled_materials`. Dead parameters only survive if this options is set (enabled by default). Setting it to `false` produces a compatible argument block layout independent of the target material mode.
- Avoid duplicate calls to common code for ternary BSDF operators and for distribution function modifiers to reduce the code size for HLSL after compilation by the `DirectXShaderCompiler` (`libbsdf`).

32.3 Fixed Bugs

32.3.1 General

- Fixed crash in scene update if a non-material is assigned as a material.
- Fixed retrieval of the geometry tag for on-demand meshes.

- Fixed crashes and data corruption when reading/writing float precision AIX files on Linux.
- Allow to start IndeX Direct without any GPU present to properly support CPU only rendering of inhomogeneous volumes.

32.3.2 Iray Photoreal

- Fixed CPU fallback in some out of memory situations.
- Fixed missing rounded corners on RTX cards (nvbugs 3235236).
- Introduced proper handling of radiance density change due to refraction effects. This makes rendering of light sources in volumes fully energy conserving. This also introduces a new option `iray_correct_radiance_on_refraction`, and it is enabled by default. Please avoid setting this to false (which is emulating the legacy behavior) as the setting may be removed in the near future again. Note though that this new setting may especially change the rendering of non-watertight meshes filled with volumes.
- Fixed bias when using multiple inhomogeneous volumes.
- Added caustic sampler connections through thin walled and invisible materials.
- Further tweak caustic sampler sampling to avoid subtle spiral patterns in some simple scenes.
- Fixed broken fallback to camera backplate in case mesh backplate function is not set.
- Fixed rendering of inhomogeneous volumes in cluster/cloud mode (nvbugs 200724792).
- Fixed texture input counting issue for emission, if cutout or textured EDF inputs along with a JIT-compiled expression is used.
- Avoid AI denoiser issues by filtering out infinite or NaN values from the auxiliary buffers (e.g. originating from JIT-compiled code).
- Added some subtle improvements/enhancements to matte object shadows.

32.3.3 Iray Interactive

- Handle invalid materials gracefully, like crashes if a non-material is assigned as a material.

32.3.4 Material Definition Language (MDL)

- Fixed wrong handling of encoded MDL names for user-defined type names and names with suffix indicating older MDL versions.
- Improved documentation and examples to demonstrate how to set the gamma mode, in particular when generating MDL source code via the module builder or when creating MDLEs.
- Fixed a crash if a reserved keyword is used as a type name.
- Fixed inlining of functions when argument expressions referenced parameters of the caller.
- Improved error reporting on broken MDL archives and MDLEs (bug 19612).
- Check upon archive creation that user-provided manifest keys form a valid identifier (bug 19612).
- Fixed creation of annotations for function variants (sometimes a wrong module name was used).
- Fixed potential crash with re-exported `enable_if()` annotations (bug 19592).

32.3.5 MI importer/exporter

- Added support for reading flags on section objects and uv projectors.
- Prevent export of (partly ancient) unsupported standard attributes on decals.

33 Iray 2021.0.0 beta, build 344800.351

33.1 Known Issues and Restrictions

- Due to the lack of CUDA capable hardware and thus also lack of a modern CUDA toolkit, CUDA acceleration is now disabled on macOS builds.
- Note that the output of normal AOV/aux buffer renderings now feature a flipped z component of the normals. This now matches the most common normal map data layouts.
- Remove legacy mia material parameter matching from thin-walled glossy BSDF handling, i.e. don't multiply exponent by '7' for thin-walled transmission. This affects `df::simple_glossy_bsdf` for modes `df::scatter_reflect_transmit` and `df::scatter_transmit` (bug 19598).
- The new heterogeneous volume support of Iray Photoreal does not yet feature a CPU backend. Thus, only GPU rendering of VDB files via IndeX Direct is supported for now. CPU support will be added with one of the next minor releases.
- The new heterogeneous volume support of Iray Photoreal does not yet support emission (via EDFs). This will be added in the next major release.

33.2 Added and Changed Features

33.2.1 General

- The Windows build is now built using VC142 (Visual Studio 2019) and the Linux builds with GCC 7.
- Minimum driver requirement (to properly support both CUDA 11.2.2 and OptiX 7.3) is 465.19.01 on Linux and 465.89 on Windows.
- Added support for Linux on Arm (including support for the NVIDIA RTX Arm PC Developer Kit). Note that support for macOS on Arm is still being worked on, while Windows on Arm is not currently planned for.
- Updated general libraries:
 - OpenSSL 1.1.1k
 - SQLite 3.34.1
 - FreeImage-3.19.x-r1859 (fixes (at least) CVE-2019-12211 and CVE-2019-12212)
 - FFmpeg 4.3.2
 - CUDA 11.2.2
 - OptiX 7.3
 - NVAPI R460
 - zlib 1.2.11 Build 339600
 - AxF 1.8.1
- Added OpenVDB support via the IndeX Direct 1.0 library.
- Added official support for caddon AIX files for spectral texture support (AIX 1.6.2).
- Added NVIDIA Texture Tools (NVTT) support to the dds plugin (e.g. to support subformats BC4-7 (with/without DirectX 10 header)).

- Added error message with details if a DDS texture can not be loaded.
- Completed support for ICanvas_opengl.
- New `backplate_mesh_function` attribute (for now limited to be picked up only by Iray Photoreal) that adds support for multiple backplates.

33.2.2 Iray API

- Added functions that allow to tessellate ffs and sds objects to ITessellator API.
- Added a parameter to `IDatabase::garbage_collection()` to control the priority of the synchronous garbage collection. Changed the default priority from `PRIORITY_LOW` to `PRIORITY_MEDIUM`.
- Deprecated support for canvases (ICanvas) with more than one tile per layer. The previous API can be re-enabled by compiling with `MI_NEURAYLIB_DEPRECATED_TILES` but note that this API will be dropped in the next major release. The factory functionality of `IImage_api` has been adapted accordingly.

33.2.3 Iray Photoreal & Iray Interactive

- Implement new (optional) high quality B-spline interpolation for bump maps (via `file_bump_texture()`, see MDL section).
- MDL 1.7 support
 - Unbounded EDF mix
 - (Color-)unbounded mix for BSDFs
 - `df::sheen_bsdf`'s multiscatter parameter
 - `df::directional_factor` for EDFs
 - `df::color*_mix`, `df::tint` and `df::unbounded_mix` for VDFs
- Improved EDF support
 - Remove the restriction that only a single EDF is supported, so now any EDF hierarchy made possible via MDL will work
 - Spectral values are now fully supported everywhere (so not just the intensity slot)
 - Texturing of EDF parameters will work now (Photoreal only, Interactive still doesn't support any texturing on lights)
- All AOV/aux buffers are rendered progressively by default now (i.e. `progressive_aux_canvas true`), to aid the quality of AI denoiser, Toon and SSIM post-processing steps. In addition a warning is emitted if it is manually disabled, but one of these post-processing steps is enabled.
- Compute preprocessing of environment importance sampling on the GPU to speed up loading/switching of environments. Note that in addition, the default baking resolution (`environment_lighting_resolution`) of the environment was increased to better match "modern" output resolutions (from 512 to 2048). For some simple scenes (e.g. objects in empty space) that rely on that default, rendering itself may become a bit slower due to that.
- Support MDL materials used as environment functions.

33.2.4 Iray Photoreal

- Support to render OpenVDB files (heterogeneous/inhomogeneous volume data) by employing the IndeX Direct library.
- Support for better volume caustics (i.e. working caustic sampler in volumes). Note that the caustic sampler still lacks connections from SSS materials to cameras placed outside the object, improvements to this will be added to the 2021.0 final.
- Faster interactive rendering on RTX GPUs (when running the interactive scheduling mode, and if no geometry motion blur is used), at the price of temporary/intermediate slightly biased results. This means that during the display of the progressive rendering results some pixel values can be slightly off, especially during the first iterations of rendering. Note though that the final image (i.e. being stopped by one of the criteria) will always be correct, same as before.
- Official spectral texture support.
- The volume stack has been extended to support priorities. An (optional) attribute `volume_priority` of type `mi::Sint8` can be set to define which object's volume properties take precedence in case of overlap.
- New "white mode", i.e. the ability to render a scene (partially) with a white diffuse material via the `iray_white_mode_enabled` boolean scene option. Note that certain material instances can be excluded from being replaced by the white material by employing the boolean attribute `exclude_from_white_mode`.
- Improved multi-tenant support
 - Added a render context option `scheduling_niceness` which works in analogy to, and via the same mechanism as, `ui_responsiveness`.
 - Extended the device throttling mechanism that is used by `ui_responsiveness` to all internal scheduling modes (e.g. when rendering larger resolutions, or all RTX GPUs in general). Note that this can lead to slightly reduced rendering performance when rendering on GPUs that drive displays by default, but at the benefit of better UI responsiveness.
 - Introduced a render context option `device_mask` for more fine grained control over device usage.
 - Introduced a render context option `cluster_partition` for more fine grained control over cluster usage.

33.2.5 Iray Interactive

- Added section capping for area lights (nvbugs 3074857).

33.2.6 Material Definition Language (MDL)

- This release changes the naming convention used for the DB elements of modules, material definitions, and function definitions. This is necessary to avoid problems that exist with the old naming scheme when names contain certain meta-characters. See `IMdl_configuration::set_encoded_names_enabled()` for details. This feature is enabled by default.
- Added the new interface `IMdl_module_builder` which allows incremental building of new MDL modules, as well as editing of existing MDL modules. This interface allows the

definition of new struct and enum types, and of new functions and materials (including variants). It also supports references to previous entities from the same module, explicit control of frequency modifiers for parameters, and parameters without defaults. An entity can also be removed from a module (if that entity is unreferenced).

The new interface can be obtained from `IMdl_factory::create_module_builder()`. See also the new `example_md1` example. The method `IMdl_factory::create_variants()` is deprecated and still available if `MI_NEURAYLIB_DEPRECATED_12_0` is defined.

- The API can be configured to treat materials as if they are simply functions with the return type `material`. This means interfaces like `IFunction_definition` and `IFunction_call` can also be used for materials. See `IMdl_configuration::set_materials_are_functions()` for details. This feature is disabled by default. It will be enabled by default in a future release.
- The new method `IMdl_factory::uniform_analysis()` allows to check the uniform property of an expression graph.
- Improved performance for loading of MDL modules, in particular parallel loading of modules, and reloading of modules.
- Added `force_default_gamma` parameter to `IMdl_impexp_api::export_canvas()` and `IExport_api::export_canvas()` to perform an automatic gamma adjustment based on the pixel type chosen for export.
- The implementation of `IFunction_definition::get_thumbnail()` and `IMaterial_definition::get_thumbnail()` has been changed to compute the value lazily on demand.
- The system locale used in methods of `IMdl_i18n_configuration` is restricted to two-letter strings to follow ISO 639-1.
- Reduced lock scope during MDL module loading. This avoids callbacks to the entity resolver and related interfaces while holding this lock.
- Added `get_md1_parameter_type_name()`, `get_return_type()` and `get_semantic()` on `IMaterial_definition` for consistency with function definition. Likewise, added `get_return_type()` on `IMaterial_instance`.
- Added `IMdl_impexp_api::get_md1_module_name()` to obtain the MDL module name for a module identified by its file name.
- Added `IType_factory::clone()` for type lists.
- The MDL compiler warns now if a literal value would loose precision in a implicit (or explicit) conversion.
- Improved half vector computation for custom-curve/measured-curve layering: assume refraction for non-thin-walled materials to loose less energy for non-physical glass materials constructed from separate BRDF and BTDF.
- Protect custom-curve evaluations against cosines > 1 to avoid numerical corner cases.
- `base.md1` now exposes a smooth B-spline interpolation mode for `base::file_bump_texture` that offers superior bump mapping at little additional runtime cost.
- Restricted several annotations to real functions (i.e. not materials): `intrinsic()`, `throws()`, `const_expr()`, `noinline()`.

- Slightly improved generated HLSL code: the HLSL optimizer can now fold constructions like `vector3(a.x, a.y, a.z)` into `a`.
- Added support for backend option "use_renderer_adapt_normal" to HLSL backend.
- `IMdl_backend::translate_environment()` accepts now a function that returns a `base::texture_return` layout-compatible type (nvbugs 200714062, bug 19608).
- Improved speed of the DAG compiler computing material hashes. (nvbugs 200700907).
- Added some mathematical identities for math functions in the DAG optimizer.
- Allowed annotation `ui_order()` on functions and materials.
- MDL 1.7 support in libbsdf:
 - Unbounded EDF mix.
 - (Color-)unbounded mix for BSDFs.
 - `df::sheen_bsdf`'s multiscatter parameter.
 - `df::directional_factor` for EDFs.
- Added support for MDL 1.7 distribution functions to the MDL distiller.
- Refactored the MDL Distiller to a plugin. Using the MDL Distiller requires now that the new plugin library `mdl_distiller.so` or `mdl_distiller.dll` is loaded with the `IPlugin_configuration::load_plugin_library()` function beforehand.

33.2.7 AxF importer

- Support for spectral color and texture data in SVBRDF, carpaint, and volumetric representations has been added.
- New AxF 1.8 SVBRDF representations with transmission color are now supported.

33.2.8 MI importer/exporter

- Added support for importing and exporting attributes of type `Ref` as `ref` rather than as `string`.
- The `.mi` file format supports a new naming scheme for MDL entities. This naming scheme avoids problems with template-like MDL functions, e.g., the array constructor. It also avoids problems when names contain certain meta-characters. Import and export of that new naming scheme requires that encoded names are enabled (see `IMdl_configuration::set_encoded_names_enabled()`). The `.mi` exporter still uses the old naming scheme by default. The exporter option `mi_mdl_new_naming_scheme` can be used to control that behavior.

33.2.9 Deep Learning based Denoiser

- Due to recent improvements in the NVIDIA driver, Iray now also uses the normal buffer to provide more information to the denoiser. This leads to better results, especially for lower sample rates/interactive usage, and detailed/small geometry.
- Improve quality of denoising in general at low sample counts for detailed/small geometry.

33.3 Fixed Bugs

33.3.1 General

- Handle case if the type of an object has changed in traversal (e.g. from a polygon mesh to a triangle mesh), which could have simply lead to a crash.
- Speed up initial scene traversal of scenes with many materials by parallelizing MDL material compilation.
- Fixed CUDA pixel type conversion path for some pixel types (e.g. PT_SINT32, PT_FLOAT32_3, and PT_FLOAT32_4).
- Significantly accelerated pixel conversion to CPU canvas (if originating from GPU).
- Fixed the rendering of camera windows on the same canvas (sometimes resetted the canvas) (nvbugs 3122569).
- Fixed (de)serialization mismatch, leading to TCPNET net errors (nvbugs 200706898).
- Properly track more of the internal CUDA memory allocations (does not include SSIM/render progress yet).
- Improved performance for semi-duplicate render target canvases (like same LPE canvas with and without alpha).
- Fixed some potential problems with updates to userdata.
- Fixed crash in Iray Bridge client if the render target has no canvas parameters.

33.3.2 Iray API

- Fixed removal of user-defined attributes (nvbugs 3119428).

33.3.3 Iray Photoreal & Iray Interactive

- More optimal/native support for the new Ampere GPUs (SM 8.6).
- Improved rendering performance for MDL JIT materials (on the average).
- Speed up transformation changes of object instances in scenes that feature a large number of materials (unless the affected objects materials in question make use of transformation matrices).
- Fixed the handling of textured inputs for the roughness of diffuse materials (nvbugs 200692801).
- Detect and output correct error messages if the wrong CUDA runtime library is used.
- Fixed handling of `math::lerp(float3, float3, float)` in the material converter.
- Fixed texture wrap mode for `df::measured_factor`.
- Fixed a crash when using a non-trivial input (e.g. function call) for a parameter of a decal projector MDL function (nvbugs 200708284).

33.3.4 Iray Photoreal

- Improved convergence while showing less temporary artifacts ('spiral' like patterns) for the caustic sampler.

- Changed handling of extinction in SSS materials/volumes if non-watertight meshes are used.
- Improved rendering of SSS materials/volumes (less temporary artifacts, better convergence).
- Fixed an issue where the ground was handled differently for volumes and surface. Now both handle the ground as solid.
- Fixed an illegal memory access when using irradiance probes (nvbugs 200698530).
- Consider motion time when setting up decal transformations.
- Employed the newly revamped RTX fiber intersector via OptiX 7.3. This leads to higher performance in most fiber heavy scenes, along with slightly improved precision.
- Slightly improved fiber geometry processing.
- Cutouts on mesh emitters are no longer ignored by parts of the renderer, leading to more correct results.
- Optimized common use case of only one material region per object, especially with sparse attribute arrays.
- Fixed some wrong error handling for OptiX Prime (i.e. pre-RTX GPUs) if CPU rendering was disabled.
- Fixed update of objects with deformation motion blur for OptiX Prime (i.e. pre-RTX GPUs) (nvbugs 200699439).
- Fixed out-of-bounds access for very low resolution 3D textures.
- Fixed update of objects with deformation motion blur on pre-RTX GPUs (nvbugs 200699439).
- Also consider backface when checking for emissive materials when updating the scene (fixes e.g. crashes/inconsistencies with auto instancing enabled when transforming emitting objects) (nvbugs 3260895).
- Improved precision for surface-varying volume coefficients and protect against negative values.

33.3.5 Iray Interactive

- Guard against malformed tangent space by the incoming geometry (e.g. normals (almost) aligned with the tangent vectors) (nvbugs 200720959).
- Made Iray Interactive consistent with Iray Photoreal's backplate behavior by using the correct background color/image/mesh when being outside a finite environment dome (nvbugs 200699016).
- Fixed invalid tag access errors for the host caches (nvbugs 200701339).
- Improved performance for some scene setups on pre-RTX GPUs.
- Fixed ray-plane intersections when the ray is parallel to the plane (nvbugs 3042132).
- Fixed handling of section capping for the UV coordinates auxiliary buffer (nvbugs 200693469).
- Added proper tracking of tonemapper changes to force a render restart (nvbugs 3268527).

33.3.6 Material Definition Language (MDL)

- Fixed `IFunction_definition::create_function_call()` for the special case of the array constructor: This function uses positional arguments. Ensure that user-supplied argument names are ignored, and instead "0", "1", and so on, are actually used.
- Fixed the methods `ILink_unit::add_material_path()` and `ILink_unit::add_material_df()` if the path involves a template-like function.
- Fixed checks for cycles in call graphs in `IMaterial_instance::create_compiled_material()`.
- Fixed the warnings about removed support for deprecation macros in `include/mi/neuraylib/version.h` to support MS Visual Studio.
- Avoid optimizations while adding to link units, making it impossible to select expression paths from certain distilled materials.
- Handle correctly auto-import of types that are used only inside struct types.
- Fixed textures with different gamma modes being reported as one texture in `ITarget_code`, when resource resolving was disabled.
- In some rare cases array constructors of kind `T[](e_1, e_2, ...)` were handled incorrectly when exported into an MDLE file. Fixed now.
- Fixed scope handling for declarations inside `then/else` and loop bodies.
- Disable tangent approximation in `::base::transform_coordinate()` to fix a performance regression (Jira OM-26192).
- Fixed a potential crash when an array size constant is used in a body of a function that gets inlined in the DAG representation.
- Fixed auxiliary base normals (libbsdf).
- Fixed handling of first component in unbounded mixers (libbsdf).
- Reduced energy loss of `df::diffuse_transmission_bsdf` as lower layer of `df::fresnel_layer`, `df::custom_curve_layer`, `df::measured_curve_layer` (libbsdf).
- Fixed incorrect contribution of `df::diffuse_transmission_bsdf` (for reflection directions) and `df::diffuse_reflection_bsdf` (for transmission directions) for evaluation with bumped normals.
- Ensure that resources cloned from another module due to a default argument are included into this module's resource table (Jira OM-23179).
- Fixed wrong function names generated for HLSL code (contained sometimes ' . ').
- Removed bogus error message if comparison of MDLEs returns inequality.
- Fixed wrong return code when setting options for LLVM-based backends.
- Use non-refracted transmission directions for non-Fresnel curve layering (libbsdf). This restores pre-2020.1.3 behavior (Jira OM-27060).
- Fixed handling of resources in reloaded modules.
- Fixed module builder such that MDL file paths are used for resources, not plain OS file names.
- Fixed `IBaker::bake_texture()` to support all pixel types for GPU baking.

- Fixed a crash when importing a function with texture-typed default arguments (bug 19589, Jira OM-23179).
- Fixed a crash when `df::tint(color, edf)` was used in some context (bug 19579).
- Fixed computation of derivation info for builtin functions that do not have a declaration (Jira OM-27580).
- Fixed code generation not restoring import tables of modules (Jira OM-23343).
- Fixed calculation of the lambda call result index in PTX backend.
- Fixed wrong `static` storage modifier for functions in HLSL (bug 19588).
- Fixed generated function names for HLSL, no more "func0" etc.
- Fixed rare code generation failure accessing the first member of a nested compound type (HLSL).
- Fixed material converter constant folding for `math::lerp()` (bug 19595).
- Fixed the pdf computation of all microfacet BSDFs in mode `df::scatter_reflect_transmit` to contain the selection probability (libbsdf).
- MDL 1.7 support
 - Unbounded EDF mix
 - (Color-)unbounded mix for BSDFs
 - `df::sheen_bsdf`'s `multiscatter` parameter
 - `df::directional_factor` for EDFs

33.3.7 AxF importer

- Optimize certain materials that feature single pixel normal maps.

33.3.8 MI importer/exporter

- Fixed an issue with very large binary vector data blocks.
- Changed `.mib` importer such that it overwrites MDL modules already existing in the database (as for any other scene element).
- Fixed `.mib` importer to handle `prefix` option correctly w.r.t. MDL elements.
- Fixed logic error during MDL parameter expression resolution that caused an endless loop (nvbugs 200697184).
- Fixed implementation of exporting individual elements (the shader decl file was missing and the `recurse` option was ignored).

34 Iray 2020.1.6, build 334300.9558

34.1 Added and Changed Features

34.1.1 General

- Updated general libraries:
 - FFmpeg 4.3.2

34.2 Fixed Bugs

34.2.1 Iray Photoreal & Iray Interactive

- Unify behavior of the backplate (between the two render modes) when leaving a finite environment dome with the camera (nvbugs 3268554 and 200713549).

35 Iray 2020.1.5, build 334300.8936

35.1 Added and Changed Features

35.1.1 General

- Updated general libraries:
 - OpenSSL 1.1.1i Build 340571
 - SQLite 3.34.1
 - FreeImage-3.19.x-r1859 (fixes (at least) CVE-2019-12211 and CVE-2019-12212)
 - FFmpeg 4.3.1 (with patches for CVE-2020-35965)

35.1.2 Iray Interactive

- Added section capping for area lights (nvbugs 3074857).

35.2 Fixed Bugs

35.2.1 Iray Photoreal & Iray Interactive

- Fixed the handling of textured inputs for the roughness of diffuse materials (nvbugs 200692801).
- Fixed a crash when using a non-trivial input (e.g. function call) for a parameter of a decal projector MDL function (nvbugs 200708284).

35.2.2 Iray Photoreal

- Fixed an illegal memory access when using irradiance probes (nvbugs 200698530).
- Fixed some wrong error handling for OptiX Prime (i.e. pre-RTX GPUs) if CPU rendering was disabled.

35.2.3 Iray Interactive

- Made Iray Interactive consistent with Iray Photoreal behavior by using the correct background color when being outside a finite environment dome (nvbugs 200699016).
- Fixed invalid tag access errors for the host caches (nvbugs 200701339).
- Fixed ray-plane intersections when the ray is parallel to the plane (nvbugs 3042132).
- Fixed handling of section capping for the UV coordinates auxiliary buffer (nvbugs 200693469).

35.2.4 Material Definition Language (MDL)

- Fixed scope handling for declarations inside then/else and loop bodies.

36 Iray 2020.1.4, build 334300.6885

36.1 Added and Changed Features

36.1.1 Iray Photoreal

- Scheduler ramp-up target is now configurable via the render context option `max_progressive_update_interval`, which controls the maximum task size that is issued to core workers. Lower values may harm efficiency (so please only tweak this value if really needed!), but increase responsiveness to interaction at later stages in the convergence process. The default remains unchanged and its recommended to leave it like this.

36.2 Fixed Bugs

36.2.1 Iray Photoreal

- Fix crash if active mBRDF elements get evicted from cache (bug 3210558)
- Fix regression/too bright SSS materials/volumes from 2020.1.2 (via the "improvements to volumes and materials featuring sub-surface-scattering") (bug 3209270)

37 Iray 2020.1.3, build 334300.6349

37.1 Added and Changed Features

37.1.1 General

- Updated general libraries:
 - Huge speedup for loading of progressive JPEGs if only the metadata is needed.
- Changed timer behavior on Windows: The May 2020 Update (20H1) of Windows 10 (build 19041) introduced a very noteworthy change to how timer resolution/accuracy is handled via the Windows API, especially in the context of many apps/processes running at the same time (for a detailed explanation see for example <https://randomascii.wordpress.com/2020/10/04/windows-timer-resolution-the-great-rule-change/>). Since Iray depends on a high timer resolution/accuracy, it now automatically calls `timeBeginPeriod(1)` on startup and `timeEndPeriod(1)` on shutdown.

37.1.2 Iray Photoreal

- Extend message tagging functionality by the ability to attach messages to all CUDA devices which adds details to the device handling. In particular, it is now possible to detect CPU fallback, or the inability to fall back to CPU.

37.1.3 Material Definition Language (MDL)

- The performance of the parallel module loading has been improved.

37.2 Fixed Bugs

37.2.1 General

- Fix crash on shutdown after failed authentication.
- Fix various race conditions potentially leading to crashes.

37.2.2 Iray API

- Fix mib import/export with prefixes.

37.2.3 Iray Photoreal & Iray Interactive

- Add message details to CUDA errors issued from POST.

37.2.4 Iray Photoreal

- Fix memory allocation error messages (Device ID and memory size swapped, bug 200677658) and details of CUDA error messages.
- Motion vectors now respect the `progressive_aux_canvas` scene option and are correctly averaged when enabled. Please note that it must be used in conjunction with `motion_vectors_instantaneous_shutter` in order to produce meaningful results (bug 3116613).
- Fix 3D textures being wrong with motion blur.

- Fix motion transform computation when instancing is on or auto for motion vectors (bug 3087345).

37.2.5 Iray Interactive

- Rework the fast convergence start mechanism that allows the iteration speed to be reduced in favor of more efficient convergence. A new scene option `irt_fast_convergence_ramp_up` has been added that allows to specify a number of frames over which the speed/convergence trade-off happens gradually. This new mechanism also fixes some issues (like render time overshooting) that the previous one had (bug 3150045).

37.2.6 Material Definition Language (MDL)

- All error messages of recursively imported modules are now reported in the list of error messages during compilation.
- The use of an imported user defined structure type with a nested structure type in an exported function signature has been fixed.
- The check for incorrect varying function call attachments to uniform parameters has been fixed in the MDL SDK API.
- The function `Mdl_compiled_material::depends_on_uniform_scenedata` has been fixed.
- The custom-curve and measured-curve layering has been improved for non-thin-walled transmissive materials to reduce energy loss for some cases of modelling glass with these components.
- The import of the `::std` module in modules of lower MDL version has been fixed.

38 Iray 2020.1.2, build 334300.5582

38.1 Added and Changed Features

38.1.1 General

- Updated general libraries:
 - OpenEXR 2.5.3

38.1.2 Material Definition Language (MDL)

- A new HLSL backend option `use_renderer_adapt_microfacet_roughness` has been added, which allows a renderer to adapt the roughness values provided to microfacet BSDFs right before using them. The prototype of the function the renderer has to provide is `float2 mdl_adapt_microfacet_roughness(Shading_state_material state, float2 roughness_uv)`.
- A new execution context option `ignore_noinline` has been added, which allows to ignore `anno::noinline()` annotations, enabling inlining when creating a compiled material. Previously this happened later when generating code for distribution functions. But optimizing at this time could lead to a changed DAG which may not contain the nodes requested by the user anymore.

38.2 Fixed Bugs

38.2.1 General

- Fix crash when editing a fiber attribute vector.
- Fix websocket bug where a fragmented message sent by a client would always be delivered flagged as a binary message.
- Fix a post-processing failure when activating/deactivating devices during rendering (bug 3108175).
- Fix post-processing crashes in a few corner cases (like no pipeline or if the pipeline runs entirely on the CPU).
- Fix potential memory leak in scheduler.
- Proper fix for scene update failing with many scene elements (bug 2505588).
- Improve performance of scene traversal for scenes that feature lots of instances.

38.2.2 Iray Photoreal & Iray Interactive

- Performance optimizations for some material paths.
- Add device IDs and tags to many more messages.
- Fix potential energy gain of multiscatter-enabled glossy BSDFs.
- Add support for tinting the environment light, driven by the scene option `environment_function_tint`.
- Fix a crash when converting materials that use colors constructed from `state::normal()` as input for normal maps (i.e. converted back to float3) (bug 3154445), but please use the new `base::blend_normals()` instead, as it will be much more efficient!

- Fix an issue where userdata lookups would fall back to the default color in JIT-compiled materials when the ground plane material changes.

38.2.3 Iray Photoreal

- If `shadow_terminator_offset_mode` is set, improve energy conservation in some corner cases, e.g. poorly tessellated geometry, notably in (but not limited to) combination with transmitting materials (example: specular BSDF with an IOR of 1.0).
- Fix an error in fiber curve-weights calculations, leading to incorrect per vertex data interpolation.
- Fix behavior of auxiliary buffers when scene features volumes, especially if the camera is inside of the media (bug 3043439, 3157501).
- General improvements to volumes and materials featuring sub-surface-scattering, both in performance/convergence and quality.
- Fix a minor issue with the light importance sampling when using userdata.
- Add experimental support for texturing volume coefficients on the surface via scene option `iray_allow_surface_volume_coefficients`.

38.2.4 Iray Interactive

- Lower CPU usage when rendering with CUDA on lower end systems.
- Fix crash if picking is done before rendering (bug 3034133).
- Fix issue with time-based termination (possible iteration overshootings, bug 3150045).

38.2.5 Material Definition Language (MDL)

- Fix wrong optimization for ternary operators selecting different vector elements in HLSL always returning the true expression.
- Fix wrong PTX version used for `sm_86`.
- In single-init mode, don't let a requested `geometry.normal` expression calculate the normal again.
- Fix analysis of derivative variants of functions not being recognized as depending on `state::normal()`.
- Reduce number of texture result slots used in generated init functions.
- Do not generate HLSL code containing `min16int` to ensure compatibility to Slang.
- Fixed translation of conversion of an 8-bit to a 32-bit integer for HLSL.
- Accept MDL projector and decal functions whose return type is explicitly marked as uniform or varying (bug 3148507).

38.2.6 MI importer/exporter

- Fixed missing error message if an MDL projector or decal function has the wrong return type.

39 Iray 2020.1.1, build 334300.4226

39.1 Added and Changed Features

39.1.1 Material Definition Language (MDL)

- Thumbnail paths are now resolved when they are requested. Before, the resolving was done during the module loading.
- A new backend option `eval_dag_ternary_strictly` has been added, which enables strict evaluation of ternary operators (?:) on the DAG to reduce code size. By default it is enabled.
- Added single-init mode for a set of functions added to a link unit, allowing all these functions to reuse values calculated in the init function and stored in the texture results field of the state struct. To enable this mode, the first path in the target function description list given to `ILink_unit::add_material()` must be "init". (Note: the init function will not be marked as `ITarget_code::DK_BSDF` anymore.)
- Improved generated code of compiled materials and lambda functions to benefit from CSE across arguments of the root node.

39.2 Fixed Bugs

39.2.1 General

- All Ampere GPUs are supported since the 2020.1.0 Beta (build 334300.1111).
- Fixed WebSocket bug in the built-in http server where fragmented web socket messages were delivered as if each fragment was a complete message. This is now fixed so that the entire message will be assembled before being delivered.
- Print an error when a non-uniform scene data lookup is attached to a uniform MDL parameter input rather than silently ignoring it.

39.2.2 Iray API

- Check for valid dimensions when creating a user data attribute vector (iray supports up to 3 components per entry, prior to this fix the API would allow an arbitrary dimension).
- Fixed crash caused by the removal of certain attributes (bug 3119428).

39.2.3 Iray Photoreal & Iray Interactive

- Fix rare problems with bump or normal maps leading to invalid results (bug 3129570).

39.2.4 Iray Photoreal

- Fix some numerical issues with section planes.
- Fix wrong motion vectors for scenes that specify an offset for the camera (bug 3111802).

39.2.5 Material Definition Language (MDL)

- Fixed `IFunction_call::get_arguments()` for the array constructor, such that it always uses "0", "1", and so on as argument names.

- Fixed failing MDLE export if the `tex::gamma_mode` type is only referenced by an annotation (bug 19551).
- Fixed storing matrices in texture results taking up all the space without much benefit.
- Fixed failure to add functions to link units when the path involves template-like functions.

39.2.6 MI importer/exporter

- User data vector import and export fixes (bug 3097124).
- Fix export of MDL material/function arguments of type string.
- Use correct argument names when adding array constructor arguments on import (bug 3089809).

40 Iray 2020.1.0, build 334300.2228

40.1 Known Issues and Restrictions

- Per-vertex userdata vectors are only supported on triangle meshes.
- MDL scene data lookups nested inside a uniform expression graph that is attached to either a uniform material slot or a uniform input of a function of the `:mdl::base` module are currently not supported (the default value will be used).
- MDL scene data lookups on materials that are attached to decals are not supported (the default value will be used).

40.2 Added and Changed Features

40.2.1 General

- Minimum driver requirement (to properly support both CUDA 11 and OptiX 7.1) is 450.51 on Linux and 451.48 on Windows.
- Switch to the CUDA 11.0.2 GA/final toolkit, and OptiX 7.1 GA/final SDK.
- Due to CUDA 11 dropping support for CentOS 6, CentOS 7 is now the minimum required version.
- Updated FFmpeg library to 4.3.1.
- Updated documentation about supported hardware and documented progress callback areas in the Programmer's Manual.
- Automatic use of CUDA for internal canvas format conversions during rendering, resulting in more interactive performance in some cases.

40.2.2 Iray API

- Added support for CUDA render targets. Introduced a new interface `IRender_target_cuda`, updated `ICanvas_cuda` to properly align with requirements. Added factory function for default implementations of `ICanvas_cuda`. This means that for Iray Interactive, data now has to potentially never leave the GPU (but (for now) only reduced copying exists in the Iray Photoreal case).

40.2.3 Material Definition Language (MDL)

- Added derivative support for matrices.
- Added derivative support for scene data functions. Requires new texture runtime functions `scene_data_lookup_deriv_float`, `scene_data_lookup_deriv_float2`, `scene_data_lookup_deriv_float3`, `scene_data_lookup_deriv_float4`, and `scene_data_lookup_deriv_color`.
- Added `mi::neuraylib::ICompiled_material::depends_on_uniform_scene_data()` analyzing whether any `scene::data_lookup_uniform_*` functions are called by a material instance.
- The return type of `IFunction_definition::get_body()` has been changed from `const IExpression_direct_call*` to `const IExpression*`.

- Implemented per function render state usage in `ITarget_code`.
- Avoid reporting deprecated warnings, if current entity is already deprecated.

40.2.4 Deep Learning based Denoiser

- Iray now uses the OptiX denoiser in the driver.

40.3 Fixed Bugs

40.3.1 General

- Added missing cloud bridge for boolean canvas parameter type.
- Fixed missing export of movable attribute for some DAG types.
- Fixed problem with toon edges on silhouettes (i.e. shared edges with the environment) (bug 3078269).
- Restore alpha channel after FXAA pass (bug 3076968).
- Fixed an issue regarding vertex user data names getting lost during mesh welding and attribute alignment.

40.3.2 Iray Photoreal & Iray Interactive

- Fix fallback behavior for currently unsupported GPUs (i.e. SM X.Y GPUs will fallback to SM X.0 then, so the major revision still has to match)
- Fix a potential numerical problem in Worley noise computation.
- Optimize performance of Perlin noise on Turing/Ampere.

40.3.3 Iray Photoreal

- Fix 2020.1 beta limitation regarding `::mdl::scene` (user data) not working for light sources.
- Further improved core memory estimates to make it more robust in corner cases (like low memory conditions).
- Fixed rendering of striped framebuffers when running under low memory conditions.
- Restored missing warning about unsupported devices, when those devices are not disabled via the NVIDIA control panel or the command line via `CUDA_VISIBLE_DEVICES`. Also made the device information less verbose, also removing some duplicate outputs.
- Reduced host memory requirements when rendering motion vectors.
- Core support for higher precision normal map encoding.
- Improve precision and robustness of fiber intersection and rendering.
- Fixed (temporary) blocky patterns appearing in volumes/SSS materials.
- Fixed special cases of new dome behavior (outside it looks black).
- Improve handling of modulated shading normals (i.e. bump and normal maps), leading to less artifacts in special cases.
- Respect object projector generated UVW coordinates in all cutout paths (bug 3068504).
- Improve handling of degenerate light geometry (i.e. zero area triangles).

- Optimize interactive performance if many LPE and/or auxiliary buffers are rendered.
- Optimize interactive performance if multiple GPUs are used.
- Optimize performance on SM 8.0/GA100 since the 2020.1 beta.
- Optimize performance of spectral rendering on Turing/Ampere.
- Fixed standard material crashes first reported in 2020.1 beta (bug 3084977).
- Fixed outliers in depth buffers (bug 3082839).
- Fixed `matte_visible_in_aux_canvas` changes appearance of result canvas (bug 3041603).
- Fixed precision issue with normal maps (banding artifacts, bug 3014665).
- Fixed potential artifacts in the caustic sampler as the scheduler cranks up iterations/update.

40.3.4 Iray Interactive

- Fix 2020.1 beta limitation regarding `::mdl::scene` (user data) not being supported.
- Fixed issues on SM 7.5/Turing and SM 8.0/Ampere GPUs when updating visibility flags.
- Fixed memory leak in host caches (bug 3072255, bug 2978402 and bug 3049585).
- Fixed incorrectly scaled environment dome if the rotation axis was not normalized.
- Fixed problem with light sources that feature same front and backside EDF (bug 3053348).
- Fixed problems with the glossy BRDF importance sampling (bug 3038834).
- Fixed race condition in texture pre-loading phase (bug 3038553 and bug 3062834).
- Optimize interactive performance if many LPE and/or auxiliary buffers are rendered.
- Fixed crash with AxF materials (bug 3074291).
- Fixed that cap color is displayed on inside surfaces of object and all internal edges become visible (bug 3049799).

40.3.5 Material Definition Language (MDL)

- Fixed checking of valid MDL identifiers (names starting with "do" were treated as keywords, but not "do" itself).
- Fixed overload resolution for MDL operators.
- Fixed compilation of materials using the array length operator (bug 19543, bug 3075690).
- Fixed crash in MDL runtime when using nonexistent image files with MDL.
- Fixed crash on CentOS 7.1 when importing non-trivial MDL modules.
- Fixed invalid translation of `int` to `float` conversion with derivatives enabled.
- Fixed broken `math::sincos()` on vectors.
- Fixed incorrect behavior during function call creation when implicit casts were enabled.
- Fixed failing MDLE creation due to several missing or non-exported entities (constants, annotations).
- Fixed failing MDLE creation if the main module was < MDL 1.6, but imported an MDL 1.6 module.
- Fixed failing MDLE creation if non-absolute imports of `::base` were used.

- Fixed rare crashes occurring when the array constructor is used in annotations.
- Fixed lost enumeration of BSDF data textures used by the libbsdf multiscatter.

40.3.6 MI importer/exporter

- Improve export of fiber data.
- Fixed export of functions from the `::<builtins>` module.
- Fixed import of calls to the array length operator in files exported before Iray 2020.
- Fixed crash when importing calls to the array length/index operator with constant expression arguments.
- Fixed crash when the parser reads an array value, but the MDL function or material parameter has no array type.

40.3.7 Deep Learning based render progress

- Fixed an issue with SSIM predictor not responding to scene changes in Iray Interactive.

41 Iray 2020.1.0 beta, build 334300.1111

41.1 Known Issues and Restrictions

- Minimum driver requirement (to properly support both CUDA 11 and OptiX 7.1) is 450.51 on Linux and 451.48 on Windows.
- The support for the `:mdl::scene` module that has been introduced with this release is so far restricted to Iray Photoreal and vertex data arrays are only working for triangle meshes. Support for other mesh types and Iray Interactive will follow.

41.2 Added and Changed Features

41.2.1 General

- Support for Ampere GPUs (SM 8.0 / GA100) in Iray Photoreal and Iray Interactive.
- Support for SM 3.X/Kepler generation GPUs has been removed due to CUDA 11.
- Replace internal OptiX 6.X based copy of the NVIDIA AI denoiser with the official OptiX 7.1 based one (via the driver). As a result, cuDNN is no longer needed by libneuray, yielding a 90 percent reduction in the size of the optimized library and avoiding JIT compilation of unneeded code on new SM versions. Also the quality and performance of denoising is improved (on the average).
- Added ability to use post SSIM as a termination criterion for Iray Photoreal and Interactive. Added a scene option `progressive_rendering_quality_ssim` to control this feature.
- Added ability to disable all post-processing for individual render target canvases via `PARAM_PROCESSING_DISABLED`. This allows to create ones own, custom post-processing pipeline for certain canvases of the rendering pass.
- The render canvas for motion vectors in screen space has been removed, now there is only one general canvas for motion vectors. See Programmers Manual section 18.2.
- The tracking of DB memory usage has been disabled by default. The tracking incurs some overhead and is only relevant if memory limits are configured or for the admin HTTP server. If needed, it can be enabled on `IDatabase_configuration`.
- Updated general libraries:
 - OpenEXR 2.5.2
 - SQLite 3.32.3
 - zlib 1.2.11
 - JsonCpp 1.9.3
 - OpenSSL 1.1.1g
 - CUDA 11.0.2
 - OptiX 7.1
 - NVAPI R445
- With this version custom attributes on scene elements of type `mi::Float32`, `mi::Float32_2`, `mi::Float32_3`, `mi::Float32_4`, `mi::Color`, `mi::Sint32`, `mi::Sint32_2`, `mi::Sint32_3` and `mi::Sint32_4` now trigger a render refresh when being created or changed unless filtered via the new functions `mi::neuraylib::IRendering_configuration::add_custom_attribute_filter()`.

41.2.2 Iray API

- The canvas type `mi::neuraylib::Canvas_type::TYPE_MOTION_VECTOR_SCREEN` has been removed.
- Custom vertex data attribute vectors are now supported via the new mesh attribute name `mi::neuraylib::ATTR_USER`.
- A new function `mi::neuraylib::IAttribute_vector::set_user_attribute_name()` has been added.
- These new `mi::neuraylib::IRendering_configuration` functions have been added:
 - `mi::neuraylib::IRendering_configuration::add_custom_attribute_filter()`
 - `mi::neuraylib::IRendering_configuration::remove_custom_attribute_filter()`
 - `mi::neuraylib::IRendering_configuration::clear_custom_attribute_filters()`
 - `mi::neuraylib::IRendering_configuration::get_custom_attribute_filter_length()`
 - `mi::neuraylib::IRendering_configuration::get_custom_attribute_filter()`
- New rectangle picking mechanism via `mi::neuraylib::IRender_context::pick()`.

41.2.3 AxF importer

- Upgraded to AxF 1.7 SDK
 - Added rudimentary support for EPSVBRDF (by utilizing automatic conversion).
 - Added support for SVBRDFs with refracting clearcoats (by utilizing the automatic conversion to non-refracting clearcoats).

41.2.4 MI importer/exporter

- Added support for MDL entities that have parenthesis and/or commas in their names. These are now escaped with a backslash to distinguish them from the syntactical elements.
- Support for reading and writing userdata arrays on `trilist` objects has been added.
- A new mi-extension allows specifying userdata names on objects in the form `name "<arrayname>" offset <index>`, where `name` denotes the name of the userdata array and `offset` specifies its index.
- The new attribute types `vector2`, `vector4`, `ivector2`, `ivector3` and `ivector4` have been introduced which map to `mi::Float32_2`, `mi::Float32_4`, `mi::Sint32_2`, `mi::Sint32_3` and `mi::Sint32_4` on the API side.

41.2.5 Iray Photoreal & Iray Interactive

- Use the newer Embree library 3.10.0 for CPU based ray tracing.
- Built-in normal/bump mapping support has been improved
 - Bump mapping functions (`base::file_texture()`, `base::tile_bump_texture()`, etc.) now support input other than `state::normal()` and `state::rounded_corner_normal()`
 - `base::blend_normals()` has been added to blend two normal maps for sticker-like use cases (base and detail normal).

- `state::rounded_corner_normal()` may now be used anywhere in the graph driving `material.geometry.normal`.
- It may be used multiple times, but only with matching parameters.
- It is still unsupported for bump inputs of layering (e.g. `df::weighted_layer`) and unsupported for JIT-compiled expressions.
- Outside of finite environment domes, rays missing the dome now return black (instead of showing the infinite environment).
- Allow total internal reflection for glossy BSDFs with mode `df::scatter_transmit`.

41.2.6 Iray Photoreal

- Extend the `iray_rt_low_memory` option (that decreases the amount of memory needed for the ray tracing acceleration hierarchies). So far this option only affected pre-Turing GPUs, but will now also save memory on the CPU.
- The controls related to motion vectors have been extended, see Programmers Manual section 4.10.
- Support for MDL scene data lookup functions has been added: up to eight userdata attributes specified on a per-vertex or per group/instance/object basis will be taken into account for scene data lookups in MDL materials.

41.2.7 Iray Interactive

- Added support for the SSIM convergence criterion.

41.2.8 Material Definition Language (MDL)

- The new API components `IMdl_impexp_api`, `IMdl_backend_api`, `IMdl_backend` and `ITarget_code` have been added to give access to the MDL compiler and related components.
- Enabled support for MDL modules whose names contains parentheses, brackets, or commas.
- The interface `IMdl_entity_resolver` has been redesigned. Support for resolving resources has been added.
- The new interface `IMdl_module_transformer` allows to apply certain transformations on MDL modules.
- Various API methods have been added in order to reduce the error-prone parsing of MDL-related names: To retrieve DB names from MDL names use `get_db_module_name()` and `get_db_definition_name()` on `IMdl_factory`. To retrieve parts of the MDL name from the corresponding DB element use `get_md1_package_component_count()`, `get_md1_package_component_name()`, and `get_md1_simple_name()` on `IModule`; `get_md1_module_name()`, `get_md1_simple_name()` on `IMaterial_definition`; and `get_md1_module_name()`, `get_md1_simple_name()`, and `get_md1_parameter_type_name()` on `IFunction_definition` and `IAnnotation_definition`.
- Added a new overload of `IModule::get_function_overloads()` that accepts a simple name and an array of parameter type names instead of two strings. This avoids the ambiguity when parsing parentheses and commas. The old overload is deprecated and still available if `MI_NEURAYLIB_DEPRECATED_11_1` is defined.

- Improved recursive MDL module reloading: changed the traversal order from pre-order to post-order traversal, avoid flagging a module as changed if it did not change at all.
- Improved `Definition_wrapper`: the creation of functions calls for template-like MDL functions requires now an actual argument list since the dummy defaults for such functions easily lead to function calls with the wrong types in the signature.
- Added more options to control the generation of compiled materials in class compilation mode: Folding of enum and bool parameters, folding of individual parameters, folding of cutout opacity, and folding of transparent layers.
- Added methods to retrieve the MDL version of modules, and the MDL version when a particular function or material definition was added to (and, if applicable, removed from) the MDL specification.
- Added methods to retrieve the MDL system and user paths.
- Changed the default MDL and resource search path: It is now empty, instead of containing the current working directory.
- Allow total internal reflection for glossy BSDFs with mode `df::scatter_transmit` (`libbsdf`).
- When derivatives are enabled, `state::position()` is now derivable. Thus, the "position" field of `Shading_state_material_with_derivs` is now a derivative type.
- Added "meters_per_scene_unit" field to `Shading_state_material`. It is used, when folding of `state::meters_per_scene_unit()` and `state::scene_units_per_meter()` has been disabled via the new `IMdl_execution_context` "fold_meters_per_scene_unit" option.
- The legacy behavior of `df::simple_glossy_bsdf` can now be controlled via the interface `IMdl_configuration`.

41.3 Fixed Bugs

41.3.1 General

- Improved SSIM convergence criterion. The previous implementation terminated once the first canvas with SSIM signaled convergence. Changed this to signal convergence only once all such canvases have converged.
- Extended integration of the SSIM pass in the post-processing pipeline to allow both SSIM and AI denoiser to be available at the same time, thus enabling and disabling is possible without restarts.
- Improved update efficiency of post-processing pipeline by avoiding unnecessary update passes if options haven't changed.

41.3.2 MDL Compiler and Backends

- Fixed serialization of ints with most significant bit set.
- Fixed file resolution during re-export of MDLE modules.
- Fixed missing clearing of context messages when creating a link unit.
- Fixed detection of absolute file names on Windows for MDLEs on a network share (bug 19517).

- Fixed support for the read-only segment and resources inside function bodies when compiling for the native target.
- Fixed rare crash/memory corruption than could occur on MDLE creation.
- The serialized code of a MDL module is now deterministic (bug 19523).
- Fixed possible crash when inlining a function containing a `for (i = ...)` loop statement (bug 19529).
- Fixed potential crash in the auto importer when imports of the current module are erroneous (Jira OM-15589).
- Fixed handling of suppressed warnings if notes are attached to them, previously these were attached to other messages.
- Fixed possible crash in generating MDLE when array types are involved (bug 19526).
- Fixed printing of initializers containing sequence expressions, it is `T v = (a, b);`, not `T v = a, b;`.
- Improved AST optimizer:
 - Write optimized `if` conditions back.
 - Write optimized sub-expressions of binary expressions back.
 - Handle constant `&& x`, constant `|| x`, `x && constant`, `x || constant`.
- Fixed documentation of `Bsdf_evaluate_data` structs: eval function results are output-only, not input/output.
- Fixed folding of calls to `state::meters_per_scene_unit()` and `state::scene_units_per_meter()` in non-inlined functions.
- Fixed wrong code generation for int to float conversions with derivatives.

41.3.3 Iray Photoreal & Iray Interactive

- `color_offset` and `color_scale` for `base::file_texture()` now also affect the mono output for mode `mono_alpha` (computed as the average of `color_offset + value * color_scale`). This restores the behavior of previous Iray versions (bug 2967468).
- Numerical issues in the computation of `base::worley_noise_bump_texture()` for `edge < 1.0` have been fixed.
- Properly handle backplate updates from multiple render contexts.

41.3.4 Iray Photoreal

- Improve precision and consistency of fiber geometry. Note that matching behavior for RTX boards is available via R450 NVIDIA drivers.
- Paths going below the ground plane are canceled, this makes the behavior between tracing and direct light estimation consistent.
- Fix temporary rendering memory handling after clearing or allocation failure.
- Fix 31 (or less) slightly wrong pixels in some scenarios on Turing GPUs.
- Don't reserve wavefront state memory before allocating framebuffer memory. The original approach favored larger wavefronts over complete framebuffers, which is unlikely to be a good tradeoff.

- Reduced risk of running out of wavefront state memory, including a fix for repeated failure to allocate wavefront state memory.
- Improve prediction of kernel launch memory, thus increasing robustness of being able to render with GPUs in low memory situations.

41.3.5 Iray Interactive

- Fix a crash when exporting stereo non-progressive buffers.
- Properly exclude hosts without GPUs from network rendering.
- Fix alpha values that should be 0 to actually be 0.
- Fix possible problems with scene updates in subsequent renders.
- Fix possible black bars appearing at the top of framebuffer.
- Decrease overhead of framebuffer copies.

41.3.6 MI importer/exporter

- Fix an issue in the mi-importer where the mesh connectivity was used for uv coordinates of polygon meshes, even though a different connectivity was required.

41.4 Iray 2020.0.3, build 327300.9514

41.5 Added and Changed Features

41.5.1 General

- Updated general libraries:
 - FFmpeg 4.3.1
 - Boost 1.69
 - OpenEXR 2.5.2
 - SQLite 3.32.3
 - zlib 1.2.11
 - JsonCpp 1.9.3
 - OpenSSL 1.1.1g
 - NVAPI R445

41.5.2 Iray Photoreal & Iray Interactive

- Use the newer Embree library 3.10.0 for CPU based ray tracing.

41.6 Fixed Bugs

41.6.1 General

- Fixed problem with toon edges on silhouettes (i.e. shared edges with the environment) (bug 3078269).
- Restore alpha channel after FXAA pass (bug 3076968).

41.6.2 Iray Photoreal & Iray Interactive

- `color_offset` and `color_scale` for `base::file_texture()` now also affect the mono output for mode `mono_alpha` (computed as the average of `color_offset + value * color_scale`). This restores the behavior of previous Iray versions (bug 2967468).

41.6.3 Iray Photoreal

- Fixed handling of negative camera offsets (bug 2978415).
- Fixed `matte_visible_in_aux_canvas` changes appearance of result canvas (bug 3041603).
- Fixed outliers in depth buffers (bug 3082839).

41.6.4 Iray Interactive

- Fixed the computation of the pixel offset when outputting non-progressive buffers (e.g. stereo) (bug 2998668).
- Fixed problems with parallel texture loading (bug 2909701, bug 3038553 and bug 3062834).
- Fixed memory leak in host caches (bug 3072255, bug 2978402 and bug 3049585).
- Fixed problems with the glossy BRDF importance sampling (bug 3038834).
- Fixed that cap color is displayed on inside surfaces of object and all internal edges become visible (bug 3049799).

41.6.5 Material Definition Language (MDL)

- Fixed serialization of ints with most significant bit set.
- Fixed possible crash when inlining a function containing a `for (i = ...)` loop statement (bug 19529).
- Fixed potential crash in the auto importer when imports of the current module are erroneous (Jira OM-15589).
- Fixed handling of suppressed warnings if notes are attached to them, previously these were attached to other messages.
- Fixed wrong code generation for int to float conversions with derivatives.

42 Iray 2020.0.2, build 327300.6313

42.1 Added and Changed Features

42.1.1 General

- Add ability to exclude objects from receiving toon edges/outlines.

42.1.2 Iray Interactive

- Add new scene option to control filtering of the implicit ground plane shadows `irt_ground_shadow_filter`. Enabled by default to match previous behavior. Could be disabled to get better AI Denoiser filtering for the ground plane.

42.1.3 Material Definition Language (MDL)

- Reduced the minimum roughness threshold for microfacet BSDFs from $1e-3$ to $1e-7$ to make them usable for mirrors and clear glass, which is inefficient but could be required by ubershaders.
- Added `"ro_data_segment"` field to `Shading_state_environment` (`"ro_data_segment_offset"` for HLSL).
- Use `"direction"` for the field name of `Shading_state_environment` (HLSL only).
- Made `state::position()` derivable.

42.2 Fixed Bugs

42.2.1 General

- Update the leaf (motion-)bounding box during on-demand mesh updates.

42.2.2 Iray Photoreal

- Fix a rare networking crash.
- Fix handling of scene changes involving multiple materials per object.
- Issue an error in cases where unavailable CUDA functionality is triggered (e.g. missing kernels).
- Better manage available GPU memory, including internal wavefront memory caches.
- Use slightly less memory for rendering on Turing GPUs.
- Fix launch memory estimate on Turing GPUs (which in one incarnation lead to the rendering of black images).
- Improve error reporting for out of memory situations.

42.2.3 Iray Interactive

- Properly handle object mask/flag updates.

42.2.4 Material Definition Language (MDL)

- Fixed some rare cases where resources inside MDL functions got lost.

- Fixed crash in MDL code generators due to MDL core compiler missing some error messages when a (wrong) member selection has the same name like an enum constant.
- Fixed rare NaN in microfacet sampling.
- Fixed error value of `ITarget_code::get_body_*` functions.
- Fixed return value of `ITarget_code::create_argument_block()` when required resource callback is missing.
- Fixed read-only data segment data not being set for native lambdas.
- Fixed resource enumeration when compiling multiple expressions in a link unit with `add_material()`: ensure that resources in material bodies are enumerated first.

42.2.5 Deep Learning based render progress - experimental feature

- Fix missing update from restarts.

43 Iray 2020.0.1, build 327300.3640

43.1 Fixed Bugs

43.1.1 General

- Fixed a bug in creating the toon post-processing pipeline.
- Improve first startup time after installing a new GPU driver (i.e. with empty/reset CUDA caches).
- Using adaptive approximation (i.e. distance ratio) of an untrimmed (freeform) surface, the whole surface instead of the limited parameter space was approximated.

43.1.2 Iray Photoreal & Iray Interactive

- Fix colored offset/scale handling for scalar textures (previously all mono-mode textures were converted to scalar data, not properly handling colored offset/scale).

43.1.3 Iray Photoreal

- Fixed some issues with material sampling, leading to improved convergence behavior (i.e. less noise on the average and improved precision in extreme cases).
- Fixed an issue with IBL/environment/dome sampling for non-power-of-two sized textures, which leads to subtle quality improvements.
- Fix potentially missing camera lens updates.
- Fix crashes when using (at least) `shadow_terminator_offset_mode` along with fiber geometry.
- Fix a problem in serialization of fiber geometry, matte area lights and animated materials (leading to artifacts like stripes in network rendering).
- Fix matte fog absorption color handling with enabled spectral rendering.

43.1.4 Iray Interactive

- Slightly improve performance on RTX GPUs in some rare scenarios.

43.1.5 Material Definition Language (MDL)

- Fixed handling of resources inside function bodies. Previously, these resources were not found under some conditions, causing black textures for instance.
- Fixed a subtle bug in one of the code caches, which caused ignored argument changes under some complex conditions. Typically, boolean parameters were vulnerable, but could happen to parameters of any type (nvbugs 2875123).
- Fixed MDL archive tool failures with Unicode package names. The MDL version of such archives is now automatically set to MDL 1.6 as lowest necessary version (bug 19512).
- A bug in the resource handling was fixed that previously caused resources to be resolved and loaded more than once, possibly leading to failures if search paths had been changed in between.

- Fixed the MDL core compiler's analysis pass. Some analysis info was computed but not annotated, causing JIT failures on functions that consists of a single expression body only.
- Fixed too strict error checks for creation of function calls of the array index operator, the ternary operator, and the cast operator.
- Fixed creation of variants without specifying any annotations where the annotations of the prototype were erroneously copied to the variants.
- Fixed loading of string-based modules with absolute file paths for resources.

44 Iray RTX 2020.0.0, build 327300.2022

Only differences to the 2020.0.0 beta will be listed here.

44.1 Known Issues and Restrictions

- Minimum driver requirement (to support CUDA 10.2) is 440.33 on Linux and 441.22 on Windows. For Turing GPUs the minimum driver requirement (to support OptiX 7.1) is 440.59 on Linux and 442.19 on Windows.
- Support for SM 3.X/Kepler generation GPUs is still only marked as deprecated, but it will most likely be removed with the next release.

44.2 Added and Changed Features

44.2.1 General

- Toon post-processing pipeline: Improve the detection, robustness, and the smoothness of lines.
- Toon post-processing pipeline: Add ability to render without input (i.e. lines (and optionally) faux shading only).
- The FreeImage plugin is now based on FreeImage 3.18.0.

44.2.2 Iray API

- A new flag on `mi::neuraylib::IMdl_configuration` instructs the MDL compiler to keep the names of let expressions and expose them as temporaries on MDL material and function definitions. This brings the structure of the material/function definition presented in the API closer to the one in the .mdl file.

44.2.3 Iray Photoreal

- Add new `iray_rt_low_memory` option to decrease the amount of memory needed for the ray tracing acceleration hierarchies. So far this option can only be set to "auto" and "on", and will only affect pre-Turing GPUs.
- Improve fiber API example and add code to demonstrate 'phantom points' (which allow B-splines to start/end at a fixed point).

44.2.4 Material Definition Language (MDL)

- Changes to the internal representation of builtin MDL operators. MDL supports a variety of operators, potentially featuring an endless number of instances:
 - array index operator `[]`
 - array length symbol
 - ternary operator `? :`

Previously, Iray created 'local' definitions for every used instance of these operators in a MDL module:

- array index operator on type T in module M: `M::T@(T, int)`
- array length symbol on type T in module M: `M::T.len(T)`

- ternary operator on type T in module M: `M::operator?(bool, T, T)`

This representation had several drawbacks:

- there might be one definition for the same operator in every module
- if the operator was not used inside the source of a module, it was not created

Especially the second point lead to several problems in the editing application. Hence, starting with the 2020.0.0 release, the internal representation was changed and operators are now represented by 'global' template-like definitions:

- array index operator: `operator[](<θ>[], int)`
- array length operator: `operator_len(<θ>[])`
- ternary operator: `operator?(bool, <θ>, <θ>)`

In addition, the name of the cast operator was changed from `operator_cast()` to `operator_cast(<θ>)`. Drawback: When inspecting the types of the operators definition, 'int' is returned for the template types, but this might be changed in the future by expanding the type system. Note: The `mi_importer` plugin will automatically convert old local operator names to the new global ones.

44.3 Fixed Bugs

44.3.1 General

- Support for processor groups on Windows, this improves utilization of all CPU cores if the system features more than 64 cores (including virtual/hyperthreaded ones).
- Fixed support for rational trim curves for free form surfaces.

44.3.2 MDL Compiler and Backends

- A bug regarding the JIT code cache was fixed. Previously, when a scene was edited and only textures or other resources were added/removed, the JIT code was not regenerated but just reused. Modifying the number or order of textures might result in different texture indexes, thus reusing the old compilation resulted in potentially wrong textures assigned, or even a GPU error.

44.3.3 Iray API

- Fixed creation of MDL function calls from array index and array length operators.

44.3.4 Iray Photoreal & Iray Interactive

- Fixes sporadic crashes, due to broken dependency handling for JIT-compiled MDL expressions that use `state::normal()`.
- Remove warning to switch from WDDM to TCC driver model if running on Optimus setups (e.g. mixed internal and discrete GPU laptops).
- Improve management of per-material texture slots to reduce dropped textures if internal texture slots are filled up.

44.3.5 Iray Photoreal

- Fixed wrong motion blur with non-infinite environment domes.
- Support textures on point light sources.

- Reduce temporary memory usage for pre-processing fibers.
- Disallow cutouts on fibers.
- Fix slightly wrong normals and v-coordinate on fibers with varying radius.
- Support negative radius values for fibers (clamped during runtime evaluation to allow for more curve effects).
- Improve robustness and precision of fiber intersector and improve self intersection ('surface acne') avoidance, especially on Turing cards.
- Fix missing intersections with fibers on pre-Turing cards.
- Fix picking of fibers on Turing cards.
- Fix minor shading issues with the new hair BSDF.
- Fix error in light source tangent handling with caustic sampler enabled.
- Fix too dark specular on ground plane.
- Fix wrong UVWs on ground plane.
- Fix negative values in depth and distance buffers for (hemi-)spherical and cylindrical camera setups.

44.3.6 Iray Interactive

- Fix a data race in render context data access/creation.
- Improved the offsetting of rays for ambient occlusion, which could show artifacts caused by self-intersection on RTX cards.
- Fix some matte fog issues.
- Fixed wrong black color rendering for section planes.
- Improve handling of failing GPUs.
- Support the new MDL 1.6 `df : : tint` modifier overload (one level of tint only).
- Fix wrong behavior of thin-walled glossy and diffuse transmission with enabled backplates.
- Improve quality of optional FXAA postprocess.
- Fix render window offset for non-progressive buffers.
- Fix handling of tangent space index for glossy BSDFs, which broke in particular MDL JIT-compiled tangent expressions.

44.3.7 Material Definition Language (MDL)

- Fix names of member selection operators/field access functions for builtin vector types. For example: `float3(float, float, float).x(float3)` is wrong, whereas `float3.x(float3)` is correct. This affects `bool`, `int`, `float`, `double` vector types.

44.3.8 MI importer/exporter

- Fix export of fibers.

44.3.9 Deep Learning based Denoiser

- Fix regression in denoising quality.

45 Iray 2020.0.0 beta, build 327300.312

45.1 Known Issues and Restrictions

- Minimum driver requirement (to support CUDA 10.2) is 440.33 on Linux and 441.22 on Windows. For Turing GPUs the minimum driver requirement (to support OptiX 7.1) is 441.87 (GA5) on Windows and on Linux the to-be-released 440 UDA 3.
- Using a driver earlier than the to-be-released 440 GA6 (i.e. 441.87 (GA5)) on Turing GPUs will result in too thin fibers.
- Support for CUDA rendering on macOS is marked as deprecated, and it will most likely be removed with the next major release.
- The new MDL 1.6 `df::tint` modifier overload is ignored so far in Iray Interactive.
- Motion vector canvas outputs are disabled for this beta, will be re-enabled for the final again.

45.2 Added and Changed Features

45.2.1 General

- Support for a new fiber primitive in order to support hair, fur and other geometries based on curves without tessellation. A new example has been added.
- Toon post-processing pipeline: This will add outlines and an (optional) faux lighting effect to a buffer output (recommended inputs are result or BSDF-weight along with anti-aliased auxiliary buffers).
- Render finish prediction. An AI network was trained to predict when rendering will be finished. This is an experimental feature. Information at which iteration rendering will be finished with a given level of quality will be provided via progress callback. For Iray Photoreal also the time needed to finish an image is returned. For quality measure SSIM (Structural Similarity Index) is used. Default is 0.98.
- Matte fog support to create fog-like atmospheric effects, such as aerial perspective, without the computational overhead of modelling an actual scattering volume.

45.2.2 Iray API

- A new scene element `mi::neuraylib::IFibers` has been added.
- The API interface `mi::neuraylib::IRender_target_base` was changed from a name based scheme to a combination of a new enum, `mi::neuraylib::Canvas_type`, and a parameter block, `mi::neuraylib::ICanvas_parameters`.
- The new API functions
 - `mi::neuraylib::IRendering_configuration::map_canvas_name()`
 - `mi::neuraylib::IRendering_configuration::map_canvas_parameters()`
 - `mi::neuraylib::IRendering_configuration::map_canvas_type()`were added to ease transition from old render target canvas names to the new interface.
- The new API functions
 - `mi::neuraylib::IModule::reload()`

- `mi::neuraylib::IModule::reload_from_string()`
- `mi::neuraylib::IModule::is_valid()`
- `mi::neuraylib::IMaterial_definition::is_valid()`
- `mi::neuraylib::IFunction_definition::is_valid()`
- `mi::neuraylib::IMaterial_instance::is_valid()`
- `mi::neuraylib::IMaterial_instance::repair()`
- `mi::neuraylib::IFunction_call::is_valid()`
- `mi::neuraylib::IFunction_call::repair()`
- `mi::neuraylib::ICompiled_material::is_valid()`

have been added to support reloading of MDL modules.

- The requirements on MDL module names have been relaxed according to the MDL 1.6 Specification to allow loading of modules with Unicode names.
- The new API functions
 - `mi::neuraylib::IMaterial_definition::get_body()`
 - `mi::neuraylib::IMaterial_definition::get_temporary_count()`
 - `mi::neuraylib::IMaterial_definition::get_temporary()`
 - `mi::neuraylib::IFunction_definition::get_body()`
 - `mi::neuraylib::IFunction_definition::get_temporary_count()`
 - `mi::neuraylib::IFunction_definition::get_temporary()`

have been added.

- The new API function

- `mi::neuraylib::IScene::set_dirty(UINT32)`

has been added, together with the declaration of a list of flags that allow the user to mark specific scene entities as dirty. At the moment only one flag can be specified (`mi::neuraylib::IScene::DIRTY_INSTANCE_TRANSFORMS`). This is useful if, as an example, animations are rendered where the camera and/or instance transforms change vastly over the course of the animation. Calling this at the beginning of each newly rendered frame can improve ray tracing precision and thus help with self intersection problems.

- The signature of the function `mi::base::ILogger::message()` has been changed.
- The API function `mi::neuraylib::ITransaction::edit()` has been adapted to disallow editing of database elements of type `mi::neuraylib::IMaterial_definition` and `mi::neuraylib::IFunction_definition`.
- Support for multiple occurrence of the same annotation has been added to `mi::neuraylib::Annotation_wrapper`.
- Support for deprecated features guarded by `MI_NEURAYLIB_DEPRECATED_8_1` and `MI_NEURAYLIB_DEPRECATED_9_1` has been removed.

45.2.3 MI importer/exporter

- The mi-importer now converts legacy MDL operator signatures to the new format.
- Support for reading and writing mental-ray-style hair objects has been added. In addition the new keyword `bspline` has been added. Note that fibers will always be exported as such.

45.2.4 Iray Photoreal & Iray Interactive

- Use the newer Embree library 3.6.1 for CPU based ray tracing.
- Support for MDL 1.6.

45.2.5 Iray Photoreal

- Support for the new fiber primitive in order to support rendering hair, fur and other geometries based on curves without tessellation.

45.2.6 Iray Interactive

- Improve CPU rendering performance.

45.2.7 Material Definition Language (MDL)

- MDL 1.6 Language Specification
 - The file path resolution algorithm has been changed to treat weak relative paths the same as strict relative paths if the referring MDL module has an MDL version of 1.6 or higher. Furthermore, the error checks have been simplified to only protect relative paths from referring to files in other search paths.
 - The import of standard library modules has been changed in all examples to use absolute path imports.
 - An additional way of defining functions has been added using an expression instead of a procedural function body.
 - Let-expression can also be applied to functions defined using an expression.
 - The limitation has been removed that package names and module names can only be identifiers.
 - The new using alias declaration has been added to enable the use of Unicode names for module names and package names.
 - The description has been clarified that standard module names shadow only modules of the same fully qualified name while modules in subpackages can have a standard module name as their unqualified name.
 - The new scene standard library module has been added with `data_isvalid`, `data_lookup_ltype`, and `data_lookup_uniform_ltype` functions.
 - The new `multiscatter_tint` parameter has been added to all glossy BSDF models to enable energy loss compensation at higher roughness values.
 - The new `df::sheen_bsdf` bidirectional scattering distribution function has been added.
 - The new `df::tint` modifier overload has been added for the hair bidirectional scattering distribution function.
 - The new `df::tint` modifier overload has been added for the separate tinting of the reflective and transmissive light paths of a base BSDF.
- Support for MDL 1.6 has been added to the MDL compiler.
- Limited support for MDL 1.6 features has been added to the backends, in particular, the scene module is supported, but currently no code is generated for interrogating the renderer, hence always the default value is returned.

- The entity resolver has been sped up for built-in modules in some cases where it is clear that the module can only be read from the MDL root.
- The memory size of the DAG representation has been slightly reduced by internalizing all DAG signatures.
- The DAG representation now uses unsafe math operations, especially $x * 0 = 0$ for floating point values.
- Inlining of functions containing constant declarations into the DAG has been implemented.
- The distiller has been extended to support the new MDL 1.6 BSDF types.
- Memory usage of identical resources occurring in different MDLEs has been reduced.

45.2.8 Deep Learning based Denoiser

- Improved quality with new training methodology, better preservation of brightness and details for low sample images.

45.3 Fixed Bugs

45.3.1 MDL Compiler and Backends

- The implementation of `math::isnan()` and `math::isfinite()` has been fixed for vector types.
- A crash in the MDL core compiler that could occur if exported types contain errors in their default initializers has been fixed.
- Wrong function names generated from `debug::assert()` calls when placed after a while loop have been fixed.
- The name of the `anno::deprecated()` parameter has been fixed, it is `description`, not `message`.
- The export of MDL modules containing relative imports has been fixed, access to the imported entities is now generated correctly.

45.3.2 Iray Photoreal & Iray Interactive

- Change behavior of the CPU fallback to be more in line with what one would expect.
- Conversion of zero-valued volume coefficient spectra for non-spectral rendering has been fixed.

45.3.3 Iray Photoreal

- Improved support and additional fixes for outputting motion vectors.
- Fix rare crashes when pre-processing the scene geometry.
- VDF mixing is now exact for up to three elemental VDFs and only approximate if more are used.

45.3.4 Iray Interactive

- Fix crash when pre-processing decal materials.
- Fix rare crashes and race conditions in the parallelized texture loading.

45.3.5 Deep Learning based Denoiser

- Fix denoising of alpha channel in denoise alpha mode on Pascal and Maxwell generation GPUs.
- Fix blend operation in denoise alpha mode.

46 Iray 2019.1.8, build 317500.18465

46.1 Added and Changed Features

46.1.1 General

- Updated FFmpeg library to 4.3.1

47 Iray 2019.1.7, build 317500.17646

47.1 Added and Changed Features

47.1.1 General

- Updated general libraries:

- Boost 1.69
- OpenEXR 2.5.2
- SQLite 3.32.3
- zlib 1.2.11
- JsonCpp 1.9.3
- OpenSSL 1.1.1g
- NVAPI R445

47.1.2 Iray Photoreal & Iray Interactive

- Use the newer Embree library 3.10.0 for CPU based ray tracing.

48 Iray 2019.1.6, build 317500.11725

48.1 Added and Changed Features

48.1.1 Iray Photoreal

- Add new `iray_rt_low_memory` option to decrease the amount of memory needed for the ray tracing acceleration hierarchies. So far this option can only be set to "auto" and "on", and will only affect pre-Turing GPUs.

48.2 Fixed Bugs

48.2.1 MDL Compiler and Backends

- A bug regarding the JIT code cache was fixed. Previously, when a scene was edited and only textures or other resources were added/removed, the JIT code was not regenerated but just reused. Modifying the number or order of textures might result in different texture indexes, thus reusing the old compilation resulted in potentially wrong textures assigned, or even a GPU error.

48.2.2 Iray Photoreal & Iray Interactive

- Fixes sporadic crashes, due to broken dependency handling for JIT-compiled MDL expressions that use `state::normal()`.
- Remove warning to switch from WDDM to TCC driver model if running on Optimus setups (e.g. mixed internal and discrete GPU laptops).

49 Iray 2019.1.5, build 317500.7473

49.1 Fixed Bugs

49.1.1 General

- Fix potential bugs due to race conditions in the scene traversal.
- Warn about non-practical pre-computed gamma corrections being done on integer format images.
- Restore forward compatibility for all non-path tracing related CUDA code (e.g. post-processing, video conversion, AI denoiser).
- Fix incorrect parsing of IES data for IESNA LM-63-2002 files.

49.1.2 MDL Compiler and Backends

- Fix slow JIT compilation of MDL code on Linux and Mac.

49.1.3 Iray Photoreal

- Fix incorrect mapping for `iray_spectral_conversion_color_space` "acescg".
- Always trigger material updates if `iray_spectral_conversion_color_space` changed.
- Fix some numerical issues.
- Work around a defect in 440 series drivers when rendering empty scenes on Turing GPUs.
- Work around a defect in 440 series drivers where initializing multiple Turing GPUs crashes.

49.1.4 Iray Interactive

- Fix that rendering was stopped after CPU load was changed.
- Fix some potential bugs with the glossy component of the ground plane.
- Proper fix for the sample queue overflow issue with specular materials and cutouts. This fixes the performance regression seen on large resolutions, but can introduce more noise in scenes with a lot of cutouts.
- Fix a potential race condition in texture pre-loading.
- Fix wrong rendering of textures after a GPU crash.
- Fix some numerical issues, specifically with spherical cameras.

50 Iray RTX 2019.1.4, build 317500.5529a

50.1 Added and Changed Features

50.1.1 General

- Update to CUDA 10.1 (Update 2) toolkit for internal compilation and the included CUDA runtime libraries. Minimum driver version requirements do not change due to this.

50.1.2 Iray API

- Two new API functions have been added to get/set the max number of allowed http connections. Default limit was increased from 100 to 256.
 - `mi::http::IServer::get_concurrent_connection_limit()`
 - `mi::http::IServer::set_concurrent_connection_limit(uint32 limit)`
- A new function `mi::neuraylib::IValue_texture::get_gamma()` has been added.
- A new function `mi::neuraylib::ICompiled_material::get_surface_opacity()` has been added.

50.1.3 MI importer/exporter

- The new options `mi_mdle_export_mode` and `mi_mdle_export_directory` have been added to the `mi_exporter`.

50.1.4 Iray Photoreal

- Add ACEScg color space support via scene option `iray_spectral_conversion_color_space`, which can now be "acescg" (in addition to "rec709", "rec2020", "xyz", and "aces"). All conversion from/to spectral supports this color space.

50.2 Fixed Bugs

50.2.1 Iray Photoreal & Iray Interactive

- More consistently handle very large and infinity values in output buffers when converting to integer formats.
- Fix implementations of anisotropic Smith masking for GGX and Beckmann.
- Fix implementation of `base::sellmeier_coefficients`.
- Fix implementation of `is_black_blend`.
- Fix crashes when requesting rendering of 0 sized images.
- Handle more MDL functions natively in the rendering core (like `math::lerp`, `math::luminance`, `float3` and color constructors) to avoid having to JIT compile MDL code that includes such functions.
- Optimize away MDL texturing nodes that are zero-weighted.

50.2.2 Iray Photoreal

- Fix light transformation (updates) regression in instancing mode "auto".

- Fix regression which would sometimes cause crashes early-on in network rendering.
- Fix incorrect stereo offsets if both stereo rendering and camera motion blur are enabled.
- Retweak self intersection handling if instancing is enabled.
- Change behavior of the "user" instancing mode. This changes "user" instancing to default to flattening (i.e. instancing being disabled).
- As a consequence, remove the "instancing" attribute (in favor of the existing "movable" attribute) to make usage consistent with the "auto" instancing mode. This yields a more predictable behavior and basically makes the "auto" instancing mode an extended "user" instancing mode.

50.2.3 Iray Interactive

- Fix crash when pre-computing the environment/IBL acceleration data.
- Fix crash at texture loading time when importing scenes.
- Respect the `progressive_rendering_max_time` more precisely.
- Fix cases where actually more iterations were computed than requested.
- Limit the number of total samples that can be computed per iteration in batch mode to avoid missing pixels or features (like when combining stereo rendering at high resolutions at high iterations).
- Improve CUDA error reporting.

50.2.4 MI importer/exporter

- The import of relative MDLE files has been fixed in the `mi_importer`.

51 Iray RTX 2019.1.3, build 317500.3714

51.1 Fixed Bugs

51.1.1 General

- Fixed removal of mesh attributes which did not work correctly if the corresponding connectivity was accessed first(bug #19213).
- Proper quantization of float to integer image formats (also fixes issue where some pixels alpha channel was rounded to 254 instead of 255 when using 8 bits per channel outputs).
- Fixed update detection of post-processing pipeline: A change in the pipeline caused the render context to request an update.
- Fixed creation of MDLE files from in-memory MDL modules.

51.1.2 Iray Photoreal & Iray Interactive

- Workaround a bug on current NVIDIA drivers that breaks ray tracing on Turing cards without RT cores.

51.1.3 Iray Photoreal

- Fixed wrong UVW coordinates generated when using some projector modes while using instancing modes "user" or "auto".
- Fixed wrong light source updates when using instancing modes "user" or "auto".
- Fixed wrong auxiliary/data buffers on Volta and Turing (e.g. when using 3D bitmap textures).
- Process the emission on double-sided materials of geometry lights as a single two-sided light source (if front- and backface emissions are identical), so these now behave the same correct way as a classical light source.
- Fixed regression of deformation/vertex motion blur being broken on Turing cards.

51.1.4 Iray Interactive

- Fix sporadic crashes when loading textures in parallel.

52 Iray 2019.1.2, build 317500.2996

52.1 Fixed Bugs

52.1.1 MDL Compiler and Backends

- A performance issue due to iterating over all files in a directory to check for non-UDIM texture files has been fixed. This fixes the loading time regression when working with a huge amount of MDL files.

52.1.2 Iray Photoreal & Iray Interactive

- Reduce GPU memory usage on RTX cards during scene pre-processing (i.e. ray tracing hierarchy construction) if instancing enabled.
- Reduce GPU memory usage on RTX cards during rendering.
- Fixed the issue of the OptiX library not being found on systems with integrated graphics or non-NVIDIA GPUs.
- Fix issue where some pixels alpha channel was rounded to 254 instead of 255 when using 8 bits per channel outputs.

52.1.3 Iray Photoreal

- Some sporadic crashes and inconsistencies fixed (includes bug LW #21668).
- Fixed wrong UVW coordinates generated when using some projector modes in combination with enabled instancing.
- Fixed wrong light source updates when using instancing modes "user" or "auto".
- Fixed wrong UVW coordinates generated when falling back to MDL JIT mode with enabled instancing.

52.1.4 Iray Interactive

- Fix black bars in high resolution images when a lot of samples/paths need to evaluate cutout opacity.
- Fix up to 15 pixel black bar on top of image when rendering with interactive scheduling and CPU-only.

52.1.5 Deep Learning based Denoiser

- If albedo and alpha buffers were used as input, the albedo image was shining through. This is fixed.