

SampleUtil Library Reference

© 2015 Sony Computer Entertainment Inc.
All Rights Reserved.
SCE Confidential

Table of Contents

Introduction.....	25
Library Summary.....	26
sce::SampleUtil.....	30
Summary	31
sce::SampleUtil	31
Functions.....	32
destroy	32
sce::SampleUtil::Audio.....	33
Summary	34
sce::SampleUtil::Audio	34
Enumerated Types	35
VoiceState	35
Type Definitions.....	36
AudioContextUpdate	36
Functions.....	37
createAudioContext.....	37
createVoiceDataFromFile.....	38
createVoiceDataFromGeneratorSetting.....	39
sce::SampleUtil::Audio::AudioContextOption.....	40
Summary	41
sce::SampleUtil::Audio::AudioContextOption	41
sce::SampleUtil::Audio::Buss	42
Summary	43
sce::SampleUtil::Audio::Buss	43
Public Instance Methods	44
setOutput.....	44
sce::SampleUtil::Audio::AudioContext	45
Summary	46
sce::SampleUtil::Audio::AudioContext	46
Public Instance Methods	47
createReverbBuss.....	47
createVoice	48
getMainBuss	49
getBgmBuss	50
sce::SampleUtil::Audio::GeneratorSettings.....	51
Summary	52
sce::SampleUtil::Audio::GeneratorSettings	52
sce::SampleUtil::Audio::VoiceData	53
Summary	54
sce::SampleUtil::Audio::VoiceData	54
sce::SampleUtil::Audio::Envelope	55
Summary	56
sce::SampleUtil::Audio::Envelope	56

sce::SampleUtil::Audio::Volume	57
Summary	58
sce::SampleUtil::Audio::Volume.....	58
sce::SampleUtil::Audio::Voice	59
Summary	60
sce::SampleUtil::Audio::Voice	60
Public Instance Methods	61
play.....	61
keyOff.....	62
kill	63
pause.....	64
resume	65
setEnvelope	66
setOutput.....	67
getState	68
setVolume	69
sce::SampleUtil::Debug	70
Summary	71
sce::SampleUtil::Debug	71
Enumerated Types	73
Type	73
Type Definitions.....	75
Callback	75
OnUpdateCallback.....	76
Functions.....	77
createConsole	77
createMenu	78
createMenuFromXmlString	79
sce::SampleUtil::Debug::Console	80
Summary	81
sce::SampleUtil::Debug::Console	81
Constructors and Destructors	82
~Console	82
Public Instance Methods	83
draw	83
vprintf.....	84
printf	85
setBgColor	86
setTextColor.....	87
enableTtyOutput.....	88
sce::SampleUtil::Debug::AbstractMenuItem.....	89
Summary	90
sce::SampleUtil::Debug::AbstractMenuItem	90
Constructors and Destructors	91
~AbstractMenuItem	91
Public Instance Methods	92
getType	92

SCE CONFIDENTIAL

getId	93
isVisible	94
setVisible	95
isActive	96
setActive	97
setCallback	98
setOnUpdateCallback	99
toString	100
update	101
update	102
sce::SampleUtil::Debug::Int8MenuItem	103
Summary	104
sce::SampleUtil::Debug::Int8MenuItem	104
Constructors and Destructors	105
~Int8MenuItem	105
Static Methods	106
downcast	106
Public Instance Methods	107
getValue	107
setValue	108
setReferenceValue	109
sce::SampleUtil::Debug::Int16MenuItem	110
Summary	111
sce::SampleUtil::Debug::Int16MenuItem	111
Constructors and Destructors	112
~Int16MenuItem	112
Static Methods	113
downcast	113
Public Instance Methods	114
getValue	114
setValue	115
setReferenceValue	116
sce::SampleUtil::Debug::Int32MenuItem	117
Summary	118
sce::SampleUtil::Debug::Int32MenuItem	118
Constructors and Destructors	119
~Int32MenuItem	119
Static Methods	120
downcast	120
Public Instance Methods	121
getValue	121
setValue	122
setReferenceValue	123
sce::SampleUtil::Debug::Int64MenuItem	124
Summary	125
sce::SampleUtil::Debug::Int64MenuItem	125
Constructors and Destructors	126

~Int64MenuItem	126
Static Methods	127
downcast	127
Public Instance Methods	128
getValue	128
setValue.....	129
setReferenceValue.....	130
sce::SampleUtil::Debug::Uint8MenuItem	131
Summary	132
sce::SampleUtil::Debug::Uint8MenuItem	132
Constructors and Destructors	133
~Uint8MenuItem.....	133
Static Methods	134
downcast	134
Public Instance Methods	135
getValue	135
setValue.....	136
setReferenceValue.....	137
sce::SampleUtil::Debug::Uint16MenuItem	138
Summary	139
sce::SampleUtil::Debug::Uint16MenuItem	139
Constructors and Destructors	140
~Uint16MenuItem.....	140
Static Methods	141
downcast	141
Public Instance Methods	142
getValue	142
setValue.....	143
setReferenceValue.....	144
sce::SampleUtil::Debug::Uint32MenuItem	145
Summary	146
sce::SampleUtil::Debug::Uint32MenuItem	146
Constructors and Destructors	147
~Uint32MenuItem.....	147
Static Methods	148
downcast	148
Public Instance Methods	149
getValue	149
setValue.....	150
setReferenceValue.....	151
sce::SampleUtil::Debug::Uint64MenuItem	152
Summary	153
sce::SampleUtil::Debug::Uint64MenuItem	153
Constructors and Destructors	154
~Uint64MenuItem.....	154
Static Methods	155
downcast	155

SCE CONFIDENTIAL

Public Instance Methods	156
getValue	156
setValue.....	157
setReferenceValue.....	158
sce::SampleUtil::Debug::FloatMenuItem.....	159
Summary	160
sce::SampleUtil::Debug::FloatMenuItem	160
Constructors and Destructors	161
~FloatMenuItem	161
Static Methods	162
downcast	162
Public Instance Methods	163
getValue	163
setValue.....	164
setReferenceValue.....	165
sce::SampleUtil::Debug::DoubleMenuItem.....	166
Summary	167
sce::SampleUtil::Debug::DoubleMenuItem	167
Constructors and Destructors	168
~DoubleMenuItem.....	168
Static Methods	169
downcast	169
Public Instance Methods	170
getValue	170
setValue.....	171
setReferenceValue.....	172
sce::SampleUtil::Debug::BoolMenuItem.....	173
Summary	174
sce::SampleUtil::Debug::BoolMenuItem	174
Constructors and Destructors	175
~BoolMenuItem.....	175
Static Methods	176
downcast	176
Public Instance Methods	177
getValue	177
setValue.....	178
setReferenceValue.....	179
sce::SampleUtil::Debug::EnumMenuItem	180
Summary	181
sce::SampleUtil::Debug::EnumMenuItem	181
Constructors and Destructors	182
~EnumMenuItem.....	182
Static Methods	183
downcast	183
Public Instance Methods	184
getValue	184
setValue.....	185

setReferenceValue	186
sce::SampleUtil::Debug::MenuItemGroup	187
Summary	188
sce::SampleUtil::Debug::MenuItemGroup	188
Constructors and Destructors	189
~MenuItemGroup	189
Static Methods	190
downcast	190
Public Instance Methods	191
getItemById	191
getItemByIndex	192
getNumItems	193
deleteItemById	194
deleteItemByIndex	195
addInt8MenuItem	196
addInt16MenuItem	197
addInt32MenuItem	198
addIntMenuItem	199
addInt64MenuItem	200
addUInt8MenuItem	201
addUInt16MenuItem	202
addUInt32MenuItem	203
addUInt64MenuItem	204
addFloatMenuItem	205
addDoubleMenuItem	206
addBoolMenuItem	207
addEnumMenuItem	208
addMenuItemGroup	209
setLabel	210
sce::SampleUtil::Debug::Menu	211
Summary	212
sce::SampleUtil::Debug::Menu	212
Constructors and Destructors	213
~Menu	213
Public Instance Methods	214
getRoot	214
findItem	215
draw	216
update	217
update	218
sce::SampleUtil::Graphics	219
Summary	220
sce::SampleUtil::Graphics	220
Enumerated Types	222
MultisampleMode	222
AttributeFormat	223
ParameterSemantic	224

SCE CONFIDENTIAL

TextureAddrMode	225
TextureFilter	226
TextureMipFilter	227
ProgramType	228
DepthFunc	229
ParameterCategory	230
ParameterType	231
CullMode	232
Primitive	233
IndexSource	234
StencilFunc	235
StencilOp	236
TextureFormat	237
ColorMask	238
BlendFunc	239
BlendFactor	240
ShaderFormat	241
BufferBindFlag	242
BufferAccessMode	243
BufferFormat	244
BufferDimension	245
FontLanguage	246
FontWeight	247
FontFamily	248
Functions	249
createGraphicsContext	249
createFontLoader	250
createSpriteRenderer	251
createObject3dRenderer	252
sce::SampleUtil::Graphics::BufferInterface	253
Summary	254
sce::SampleUtil::Graphics::BufferInterface	254
Public Instance Methods	255
beginWrite	255
endWrite	256
getLoader	257
getDimension	258
sce::SampleUtil::Graphics::Buffer	259
Summary	260
sce::SampleUtil::Graphics::Buffer	260
Public Instance Methods	261
beginWrite	261
endWrite	262
getLoader	263
getSize	264
getBindFlags	265
getAccessMode	266
getElementSize	267

sce::SampleUtil::Graphics::IndexBuffer.....	268
Summary	269
sce::SampleUtil::Graphics::IndexBuffer	269
Protected Instance Methods	270
getBaseBuffer	270
sce::SampleUtil::Graphics::VertexBuffer	271
Summary	272
sce::SampleUtil::Graphics::VertexBuffer	272
Protected Instance Methods	273
getBaseBuffer	273
sce::SampleUtil::Graphics::UniformBuffer	274
Summary	275
sce::SampleUtil::Graphics::UniformBuffer	275
Protected Instance Methods	276
getBaseBuffer	276
sce::SampleUtil::Graphics::StructuredBuffer.....	277
Summary	278
sce::SampleUtil::Graphics::StructuredBuffer.....	278
Protected Instance Methods	279
getBaseBuffer	279
sce::SampleUtil::Graphics::Texture	280
Summary	281
sce::SampleUtil::Graphics::Texture	281
Public Instance Methods	282
setAddrMode	282
setFilter	283
getWidth	284
getHeight	285
getStride	286
sce::SampleUtil::Graphics::RenderTarget	287
Summary	288
sce::SampleUtil::Graphics::RenderTarget	288
Public Instance Methods	289
getWidth	289
getHeight	290
getTexture	291
sce::SampleUtil::Graphics::DepthStencilSurface	292
Summary	293
sce::SampleUtil::Graphics::DepthStencilSurface	293
Public Instance Methods	294
getTexture	294
sce::SampleUtil::Graphics::Collada::ColladaData	295
Summary	296
sce::SampleUtil::Graphics::Collada::ColladaData	296
Public Instance Methods	297
getLoader	297

getImageById	298
getImageByIndex	299
getImageByIndex	300
getNumImages	301
getMaterialById	302
getMaterialByIndex	303
getNumMaterials	304
getMeshByControllerId	305
getMeshByGeometryId	306
getMeshByIndex	307
getNumMeshes	308
getVisualScene	309
getAnimation	310
sce::SampleUtil::Graphics::Collada::NodeTransform	311
Summary	312
sce::SampleUtil::Graphics::Collada::NodeTransform	312
sce::SampleUtil::Graphics::Collada::NodeAnimation	313
Summary	314
sce::SampleUtil::Graphics::Collada::NodeAnimation	314
Public Instance Methods	315
getNodeId	315
getNumFrames	316
getNodeTransform	317
getMatrix	318
sce::SampleUtil::Graphics::Collada::ShaderParameterReference	319
Summary	320
sce::SampleUtil::Graphics::Collada::ShaderParameterReference	320
Constructors and Destructors	321
~ShaderParameterReference	321
Public Instance Methods	322
getParamValue	322
sce::SampleUtil::Graphics::Collada::MatrixChangeListener	323
Summary	324
sce::SampleUtil::Graphics::Collada::MatrixChangeListener	324
Constructors and Destructors	325
~MatrixChangeListener	325
Public Instance Methods	326
setWorldMatrix	326
sce::SampleUtil::Graphics::Collada::DefaultShaderParameterManager	327
Summary	328
sce::SampleUtil::Graphics::Collada::DefaultShaderParameterManager	328
Public Instance Methods	329
setLight	329
setWorldMatrix	330
setViewMatrix	331
setProjectionMatrix	332
setShadowTex	333

setShadowViewProjection.....	334
sce::SampleUtil::Graphics::Collada::ColladaLoader.....	335
Summary	336
sce::SampleUtil::Graphics::Collada::ColladaLoader	336
Public Instance Methods	337
load	337
getDefaultValue	338
getGraphicsLoader.....	339
getMatrixChangeListener.....	340
sce::SampleUtil::Graphics::Collada::EffectDataParameter.....	341
Summary	342
sce::SampleUtil::Graphics::Collada::EffectDataParameter.....	342
Enumerated Types	343
Type	343
Constructors and Destructors	344
EffectDataParameter.....	344
EffectDataParameter.....	345
EffectDataParameter.....	346
EffectDataParameter.....	347
~EffectDataParameter.....	348
Operator Methods	349
operator=.....	349
Public Instance Methods	350
getName.....	350
getType	351
getFloatArray.....	352
getFloatArraySize.....	353
getImageId	354
sce::SampleUtil::Graphics::Collada::EffectDataShader.....	355
Summary	356
sce::SampleUtil::Graphics::Collada::EffectDataShader	356
Public Instance Methods	357
getPath	357
getEntry	358
getNumParameters	359
addParameter	360
sce::SampleUtil::Graphics::Collada::EffectData	361
Summary	362
sce::SampleUtil::Graphics::Collada::EffectData.....	362
Public Instance Methods	363
getVertexShader	363
getFragmentShader	364
getVertexShader	365
getFragmentShader	366
sce::SampleUtil::Graphics::Collada::Material.....	367
Summary	368
sce::SampleUtil::Graphics::Collada::Material	368

SCE CONFIDENTIAL

Public Instance Methods	369
getId	369
getEffect	370
getEffectData.....	371
sce::SampleUtil::Graphics::Collada::MeshPart.....	372
Summary	373
sce::SampleUtil::Graphics::Collada::MeshPart.....	373
Public Instance Methods	374
getVertexAttribute.....	374
getVertexAttribute.....	375
getNumVertexAttributes	376
getStride	377
getVertexBuffer	378
getIndexBuffer.....	379
getNumIndices	380
getNumVertices.....	381
getMaterial	382
hasSkin	383
sce::SampleUtil::Graphics::Collada::Mesh.....	384
Summary	385
sce::SampleUtil::Graphics::Collada::Mesh.....	385
Enumerated Types	386
JointNameType	386
Public Instance Methods	387
hasSkin	387
getMeshPart.....	388
getNumMeshParts	389
getControllerId.....	390
getGeometryId	391
getJointNameType	392
getNumJoints	393
getJointName	394
getInvBindMatrices.....	395
getBindShapeMatrix.....	396
sce::SampleUtil::Graphics::Collada::Image.....	397
Summary	398
sce::SampleUtil::Graphics::Collada::Image	398
Public Instance Methods	399
getTexture	399
getTexture	400
getId	401
getName.....	402
getAbsolutePath.....	403
getPath	404
sce::SampleUtil::Graphics::Collada::InstanceMeshPart	405
Summary	406
sce::SampleUtil::Graphics::Collada::InstanceMeshPart	406

SCE CONFIDENTIAL

Public Instance Methods	407
draw	407
applyShaderParameterManager	408
addEffect	409
sce::SampleUtil::Graphics::Collada::InstanceMesh	410
Summary	411
sce::SampleUtil::Graphics::Collada::InstanceMesh	411
Public Instance Methods	412
updateSkinMatrices	412
applyShaderParameterManager	413
draw	414
getNumMeshParts	415
getMeshPart	416
getMesh	417
addEffect	418
sce::SampleUtil::Graphics::Collada::Node	419
Summary	420
sce::SampleUtil::Graphics::Collada::Node	420
Public Instance Methods	421
getId	421
getSid	422
findNodeBySid	423
findNodeById	424
findNodeBySid	425
findNodeById	426
getTransformMatrix	427
setTransformMatrix	428
getWorldMatrix	429
getWorldMatrixPointer	430
updateWorldMatrix	431
updateSkinMatrices	432
draw	433
applyShaderParameterManager	434
getInstanceMesh	435
getNumInstanceMeshes	436
getNumChildNodes	437
getChildNode	438
getChildNode	439
addEffect	440
sce::SampleUtil::Graphics::Collada::Animation	441
Summary	442
sce::SampleUtil::Graphics::Collada::Animation	442
Public Instance Methods	443
getStartTime	443
getEndTime	444
getNumFrames	445
getNumAnimationNodes	446

getAnimationNode.....	447
sce::SampleUtil::Graphics::Collada::VisualScene	448
Summary	449
sce::SampleUtil::Graphics::Collada::VisualScene	449
Public Instance Methods	450
getId	450
getName.....	451
findNodeById.....	452
findNodeBySid	453
findNodeById.....	454
findNodeBySid	455
get rootNode	456
get rootNode	457
sce::SampleUtil::Graphics::Collada::InstanceVisualScene	458
Summary	459
sce::SampleUtil::Graphics::Collada::InstanceVisualScene.....	459
Public Instance Methods	460
get rootNode	460
get rootNode	461
draw	462
applyParameterReference	463
addEffect	464
findNodeById.....	465
findNodeBySid	466
findNodeById.....	467
findNodeBySid	468
getVisualScene	469
sce::SampleUtil::Graphics::Collada::AnimationPlayer.....	470
Summary	471
sce::SampleUtil::Graphics::Collada::AnimationPlayer	471
Public Instance Methods	472
setTime.....	472
setFrame	473
get Animation	474
sce::SampleUtil::Graphics::BlendInfo	475
Summary	476
sce::SampleUtil::Graphics::BlendInfo	476
Public Instance Methods	477
initializeAsNoBlend	477
initializeAsAlphaBlend.....	478
initializeAsAddBlend.....	479
sce::SampleUtil::Graphics::GraphicsContextOption.....	480
Summary	481
sce::SampleUtil::Graphics::GraphicsContextOption	481
Constructors and Destructors	482
GraphicsContextOption	482
sce::SampleUtil::Graphics::GraphicsContext.....	483

Summary	484
sce::SampleUtil::Graphics::GraphicsContext.....	484
Enumerated Types	485
FrameBufferSide	485
DisplayMode	486
Public Instance Methods	487
clearRenderTarget.....	487
isDisplaymodeSupported	488
setDisplaymode.....	489
getNextRenderTarget.....	490
getDepthStencilSurface	491
beginScene	492
endScene	493
flip.....	494
setDepthFunc.....	495
setDepthWriteEnable	496
setFillMode.....	497
setCullMode	498
setVertexProgram	499
setFragmentProgram	500
setVertexBuffer.....	501
draw	502
setUniformBuffer	503
setTexture.....	504
reserveDefaultUniformBuffer.....	505
setViewPort	506
setRegionClip	507
setStencilFunc.....	508
getCurrentRenderTarget	509
getCurrentDepthStencilSurface	510
setLineWidth	511
saveBackBufferAsBmp	512
sce::SampleUtil::Graphics::Effect.....	513
Summary	514
sce::SampleUtil::Graphics::Effect	514
sce::SampleUtil::Graphics::Effect::ParameterValue.....	515
Summary	516
sce::SampleUtil::Graphics::Effect::ParameterValue.....	516
Constructors and Destructors	517
ParameterValue	517
~ParameterValue	518
Public Instance Methods	519
initializeAsArray.....	519
initializeAsTexture	520
setArrayDataF	521
setArrayData	522
setTexture.....	523
finalize	524

SCE CONFIDENTIAL

getCategory.....	525
getType	526
getData.....	527
getData.....	528
getTexture	529
copyFrom	530
sce::SampleUtil::Graphics::Effect::EffectParameter	531
Summary	532
sce::SampleUtil::Graphics::Effect::EffectParameter	532
Constructors and Destructors	533
EffectParameter	533
~EffectParameter	534
Public Instance Methods	535
initialize	535
getParameter	536
getValue	537
getValue	538
setToUniformBuffer	539
finalize	540
copyFrom	541
getProgramType.....	542
sce::SampleUtil::Graphics::Effect::EffectShader.....	543
Summary	544
sce::SampleUtil::Graphics::Effect::EffectShader.....	544
Constructors and Destructors	545
EffectShader	545
~EffectShader	546
Public Instance Methods	547
initialize	547
getNumParams	548
getParamIndex.....	549
getParamByIndex.....	550
getParamByName.....	551
getParamByIndex.....	552
getParamByName.....	553
getProgramId	554
finalize	555
copyFrom	556
sce::SampleUtil::Graphics::Effect::EffectData	557
Summary	558
sce::SampleUtil::Graphics::Effect::EffectData.....	558
Constructors and Destructors	559
EffectData.....	559
~EffectData	560
Public Instance Methods	561
initialize	561
finalize	562

SCE CONFIDENTIAL

getVertexShader	563
getFragmentShader	564
getVertexShader	565
getFragmentShader	566
getNumAttributeParams.....	567
getAttributeParam	568
copyFrom	569
sce::SampleUtil::Graphics::Effect::EffectInstance.....	570
Summary	571
sce::SampleUtil::Graphics::Effect::EffectInstance.....	571
Constructors and Destructors	572
EffectInstance.....	572
~EffectInstance	573
Public Instance Methods	574
initialize	574
finalize	575
getEffect	576
setParamValueReferenceByName	577
apply.....	578
cloneFrom	579
sce::SampleUtil::Graphics::FontParam.....	580
Summary	581
sce::SampleUtil::Graphics::FontParam.....	581
Public Instance Methods	582
setDefaults	582
sce::SampleUtil::Graphics::Font.....	583
Summary	584
sce::SampleUtil::Graphics::Font	584
Public Instance Methods	585
cacheCharacters	585
clearCache	586
sce::SampleUtil::Graphics::FontLoader.....	587
Summary	588
sce::SampleUtil::Graphics::FontLoader.....	588
Public Instance Methods	589
createFont	589
sce::SampleUtil::Graphics::VertexProgram.....	590
Summary	591
sce::SampleUtil::Graphics::VertexProgram	591
sce::SampleUtil::Graphics::FragmentProgram	592
Summary	593
sce::SampleUtil::Graphics::FragmentProgram	593
sce::SampleUtil::Graphics::VertexBuffer.....	594
Summary	595
sce::SampleUtil::Graphics::VertexBuffer	595
sce::SampleUtil::Graphics::VertexAttrib.....	596

SCE CONFIDENTIAL

Summary	597
sce::SampleUtil::Graphics::VertexAttribute	597
sce::SampleUtil::Graphics::GraphicsLoader	598
Summary	599
sce::SampleUtil::Graphics::GraphicsLoader	599
Public Instance Methods	600
registerVertexProgram	600
registerFragmentProgram	601
createVertexProgram	602
createFragmentProgram	603
cloneVertexProgram	604
cloneFragmentProgram	605
createIndexBuffer	606
createVertexBuffer	607
createUniformBuffer	608
createBuffer	609
createTexture2dBuffer	610
createRenderTarget	611
createDepthStencilSurface	612
createTextureFromFile	613
createTextureFromMemory	614
createTexture	615
sce::SampleUtil::Graphics::Gxm::CommonDialogUpdateParam	616
Summary	617
sce::SampleUtil::Graphics::Gxm::CommonDialogUpdateParam	617
Constructors and Destructors	618
CommonDialogUpdateParam	618
Public Instance Methods	619
initialize	619
finalize	620
sce::SampleUtil::Graphics::Parameter	621
Summary	622
sce::SampleUtil::Graphics::Parameter	622
Public Instance Methods	623
getSemantic	623
getComponentCount	624
getArraySize	625
isUseDefaultBuf	626
getResourceIndex	627
getContainerIndex	628
getSemanticIndex	629
setUniformDataF	630
getName	631
getCategory	632
isValid	633
getType	634
sce::SampleUtil::Graphics::Program	635

SCE CONFIDENTIAL

Summary	636
sce::SampleUtil::Graphics::Program.....	636
Public Instance Methods	637
isValid	637
getParameterCount.....	638
findParameterByName.....	639
findParameterBySemantic	640
getParameter	641
sce::SampleUtil::Graphics::SpriteRenderer.....	642
Summary	643
sce::SampleUtil::Graphics::SpriteRenderer	643
Public Instance Methods	644
setRenderTargetSize.....	644
fillRect	645
drawRect	646
fillOval	647
drawOval	648
drawLine.....	649
drawTexture.....	650
drawTexture.....	651
drawTextureYuy2.....	652
drawTextureYuy2.....	653
drawPoints	654
drawString	655
getStringTextureSize	656
drawDebugString.....	657
drawDebugString.....	658
drawDebugStringf.....	659
drawDebugStringf.....	660
getWidthOfDebugChar	661
sce::SampleUtil::Graphics::Object3dRenderer	662
Summary	663
sce::SampleUtil::Graphics::Object3dRenderer	663
Public Instance Methods	664
fillCube	664
drawCube	665
drawLine.....	666
drawSphere	667
fillSphere	668
fillCylinder.....	669
drawCylinder	670
sce::SampleUtil::Input::PadContextOption.....	671
Summary	672
sce::SampleUtil::Input::PadContextOption	672
Constructors and Destructors	673
PadContextOption.....	673
sce::SampleUtil::Input::TouchPadData.....	674

SCE CONFIDENTIAL

Summary	675
sce::SampleUtil::Input::TouchPadData	675
sce::SampleUtil::Input::MagnetometerData	676
Summary	677
sce::SampleUtil::Input::MagnetometerData	677
sce::SampleUtil::Input::MotionData	678
Summary	679
sce::SampleUtil::Input::MotionData	679
sce::SampleUtil::Input::PadData	680
Summary	681
sce::SampleUtil::Input::PadData	681
sce::SampleUtil::Input::PadContext	682
Summary	683
sce::SampleUtil::Input::PadContext	683
Constructors and Destructors	684
~PadContext	684
Public Instance Methods	685
update	685
isButtonDown	686
isButtonUp	687
isButtonPressed	688
isButtonReleased	689
getLeftStick	690
getRightStick	691
setRightAnalogStickDeadZone	692
setLeftAnalogStickDeadZone	693
getData	694
getPreviousUpdateData	695
enableMotionSensor	696
enableMagnetometer	697
enableTiltCorrection	698
enableAngularVelocityDeadband	699
resetOrientation	700
getDefaultValueLeftAnalogStickDeadZone	701
getDefaultValueRightAnalogStickDeadZone	702
sce::SampleUtil::Input	703
Summary	704
sce::SampleUtil::Input	704
Enumerated Types	705
ButtonEventPattern	705
Button	706
MotionContextFunctionFlag	707
TouchPort	708
TouchContextFunctionFlag	709
Functions	710
createPadContext	710
createControllerContext	711

SCE CONFIDENTIAL

createMotionContext	712
createTouchContext	713
sce::SampleUtil::Input::ControllerContextOption	714
Summary	715
sce::SampleUtil::Input::ControllerContextOption	715
sce::SampleUtil::Input::ControllerContext	716
Summary	717
sce::SampleUtil::Input::ControllerContext	717
Public Instance Methods	718
update	718
isButtonDown	719
isButtonUp	720
isButtonPressed	721
isButtonReleased	722
getLeftStick	723
getRightStick	724
setDeadZone	725
getData	726
sce::SampleUtil::Input::MotionContextData	727
Summary	728
sce::SampleUtil::Input::MotionContextData	728
sce::SampleUtil::Input::MotionContext	729
Summary	730
sce::SampleUtil::Input::MotionContext	730
Public Instance Methods	731
update	731
enableFunctions	732
getEnabledFunctions	733
getMotionState	734
getMotionContextData	735
resetReferenceOrientation	736
rotateReferenceOrientationYaw	737
setUserInterfaceOrientationThreshold	738
enableAccelerometerTiltCorrection	739
enableGyroDeadband	740
enableGyroBiasCorrection	741
sce::SampleUtil::Input::TouchContextOption	742
Summary	743
sce::SampleUtil::Input::TouchContextOption	743
sce::SampleUtil::Input::TouchPrimitiveGestureEvent	744
Summary	745
sce::SampleUtil::Input::TouchPrimitiveGestureEvent	745
sce::SampleUtil::Input::TouchGestureEvent	746
Summary	747
sce::SampleUtil::Input::TouchGestureEvent	747
sce::SampleUtil::Input::TouchContext	748

Summary	749
sce::SampleUtil::Input::TouchContext	749
Public Instance Methods	750
update	750
setFunctionFlag	751
getFunctionFlag	752
getRawData	753
getPanelInfo	754
getNumberOfPrimitiveGestureEvents	755
getPrimitiveGestureEvents	756
getNumberOfGestureEvents	757
getGestureEvents	758
sce::SampleUtil::Resource	759
Summary	760
sce::SampleUtil::Resource	760
sce::SampleUtil::SampleSkeleton	761
Summary	762
sce::SampleUtil::SampleSkeleton	762
Enumerated Types	763
SampleSkeletonFunctionFlag	763
ControllerFunctionContextMode	765
Constructors and Destructors	766
SampleSkeleton	766
~SampleSkeleton	767
Public Instance Methods	768
initialize	768
update	769
render	770
finalize	771
Protected Instance Methods	772
initializeUtil	772
updateUtil	773
finalizeUtil	774
renderDebugText	775
getUserIdManager	776
getGraphicsContext	777
getControllerContext	778
getPadContextOfInitialUser	779
getSpriteRenderer	780
getMotionContext	781
getTouchContext	782
getAudioContext	783
sce::SampleUtil::SampleSkeleton::SampleSkeletonOption	784
Summary	785
sce::SampleUtil::SampleSkeleton::SampleSkeletonOption	785
Constructors and Destructors	786
SampleSkeletonOption	786

sce::SampleUtil::System	787
Summary	788
sce::SampleUtil::System	788
Enumerated Types	789
UserColor	789
SaveDataSlotStatus	790
SaveDataSlotSortKey	791
SaveDataSlotSortOrder	792
SaveDataSaveMode	793
Type Definitions	794
UserId.....	794
UserIdManagerOption.....	795
SaveDataSlotId	796
SaveDataSlotParam.....	797
SaveDataSearchCond	798
SaveDataSearchResult.....	799
DataSaveItem	800
SaveDataMountPoint	801
DataRemoveItem	802
SaveDataOption	803
Functions	804
createUserIdManager	804
createSaveData	805
sce::SampleUtil::System::UserIdList	806
Summary	807
sce::SampleUtil::System::UserIdList	807
sce::SampleUtil::System::UserLoginStatusChangedEvents	808
Summary	809
sce::SampleUtil::System::UserLoginStatusChangedEvents	809
sce::SampleUtil::System::UserIdManager	810
Summary	811
sce::SampleUtil::System::UserIdManager	811
Constructors and Destructors	812
~UserIdManager	812
Public Instance Methods	813
update	813
getLoginUserIdList	814
getUserLoginStatusChangedEventsOfLastUpdate	815
getInitialUser	816
getUserName	817
getUserColor	818
sce::SampleUtil::System::UserIdManagerOption	819
Summary	820
sce::SampleUtil::System::UserIdManagerOption	820
sce::SampleUtil::System::SaveDataSlotParam	821
Summary	822
sce::SampleUtil::System::SaveDataSlotParam	822

sce::SampleUtil::System::SaveDataSearchCond	823
Summary	824
sce::SampleUtil::System::SaveDataSearchCond	824
sce::SampleUtil::System::SaveDataSearchResult	825
Summary	826
sce::SampleUtil::System::SaveDataSearchResult	826
sce::SampleUtil::System::DataSaveItem	827
Summary	828
sce::SampleUtil::System::DataSaveItem	828
sce::SampleUtil::System::SaveDataMountPoint	829
Summary	830
sce::SampleUtil::System::SaveDataMountPoint	830
sce::SampleUtil::System::DataRemoveItem	831
Summary	832
sce::SampleUtil::System::DataRemoveItem	832
sce::SampleUtil::System::SaveData	833
Summary	834
sce::SampleUtil::System::SaveData	834
Public Instance Methods	835
slotCreate	835
slotDelete	836
slotSetParam	837
slotGetParam	838
slotSearch	839
dataSave	840
atomicSlotCreateAndDataSave	841
atomicDataSaveAndSlotSetParam	842
dataLoadMount	843
dataLoadUmount	844
dataRemove	845
sce::SampleUtil::System::SaveDataOption	846
Summary	847
sce::SampleUtil::System::SaveDataOption	847

SCE CONFIDENTIAL

Introduction

Library Summary

Library Contents

Item	Description
sce::SampleUtil	Sample utility.
sce::SampleUtil::Audio	Audio-associated definitions.
sce::SampleUtil::Audio::AudioContextOption	AudioContext option.
sce::SampleUtil::Audio::Buss	Buss class.
sce::SampleUtil::Audio::AudioContext	Context class for playing a sound.
sce::SampleUtil::Audio::GeneratorSettings	Parameter used for the initialization of signal generator format voice data.
sce::SampleUtil::Audio::VoiceData	Voice data type.
sce::SampleUtil::Audio::Envelope	Structure used to set an envelope.
sce::SampleUtil::Audio::Volume	Structure used to set a volume value.
sce::SampleUtil::Audio::Voice	Class to provide the voice playing back function.
sce::SampleUtil::Debug	Debug-associated definitions.
sce::SampleUtil::Debug::Console	Class that handles the debug console.
sce::SampleUtil::Debug::AbstractMenuItem	Base class of each menu item class.
sce::SampleUtil::Debug::Int8MenuItem	Debug menu item class that can manage signed 8-bit integer type values.
sce::SampleUtil::Debug::Int16MenuItem	Debug menu item class that can manage signed 16-bit integer type values.
sce::SampleUtil::Debug::Int32MenuItem	Debug menu item class that can manage signed 32-bit integer type values.
sce::SampleUtil::Debug::Int64MenuItem	Debug menu item class that can manage signed 64-bit integer type values.
sce::SampleUtil::Debug::Uint8MenuItem	Debug menu item class that can manage unsigned 8-bit integer type values.
sce::SampleUtil::Debug::Uint16MenuItem	Debug menu item class that can manage unsigned 16-bit integer type values.
sce::SampleUtil::Debug::Uint32MenuItem	Debug menu item class that can manage unsigned 32-bit integer type values.
sce::SampleUtil::Debug::Uint64MenuItem	Debug menu item class that can manage unsigned 64-bit integer type values.
sce::SampleUtil::Debug::FloatMenuItem	Debug menu item class that can manage single precision floating-point type values.

SCE CONFIDENTIAL

Item	Description
sce::SampleUtil::Debug::DoubleMenuItem	Debug menu item class that can manage double precision floating-point type values.
sce::SampleUtil::Debug::BoolMenuItem	Debug menu item class that can manage Bool type values.
sce::SampleUtil::Debug::EnumMenuItem	Debug menu item class that can manage indexed character string set types.
sce::SampleUtil::Debug::MenuItemGroup	Debug menu item class that can group multiple AbstractMenuItems.
sce::SampleUtil::Debug::Menu	Class that performs management/rendering/etc. for a menu that is a tree structure.
sce::SampleUtil::Graphics	Graphics-associated definitions.
sce::SampleUtil::Graphics::BufferInterface	Class which becomes the base of a buffer object.
sce::SampleUtil::Graphics::Buffer	One-dimensional buffer class.
sce::SampleUtil::Graphics::IndexBuffer	Index buffer.
sce::SampleUtil::Graphics::VertexBuffer	Vertex buffer.
sce::SampleUtil::Graphics::UniformBuffer	Uniform buffer.
sce::SampleUtil::Graphics::StructuredBuffer	Structured buffer.
sce::SampleUtil::Graphics::Texture	Texture buffer class.
sce::SampleUtil::Graphics::RenderTarget	Render target.
sce::SampleUtil::Graphics::DepthStencilSurface	Depth stencil target.
sce::SampleUtil::Graphics::Collada::ColladaData	Class of the Collada data.
sce::SampleUtil::Graphics::Collada::NodeTransform	Transform of a node in a scene.
sce::SampleUtil::Graphics::Collada::NodeAnimation	Animation data of one node.
sce::SampleUtil::Graphics::Collada::ShaderParameterReference	The interface for a class to refer to a shader parameter.
sce::SampleUtil::Graphics::Collada::MatrixChangeListener	Interface to notify the World matrix when drawing a node.
sce::SampleUtil::Graphics::Collada::DefaultShaderParameterManager	Default shader parameter manager.
sce::SampleUtil::Graphics::Collada::ColladaLoader	Manages the data shared by multiple Colladas.
sce::SampleUtil::Graphics::Collada::EffectDataParameter	Parameter of an effect.
sce::SampleUtil::Graphics::Collada::EffectDataShader	The shader information of an effect.
sce::SampleUtil::Graphics::Collada::EffectData	Effect data.
sce::SampleUtil::Graphics::Collada::Material	Material class.
sce::SampleUtil::Graphics::Collada::MeshPart	Mesh part class.
sce::SampleUtil::Graphics::Collada::Mesh	Mesh class.
sce::SampleUtil::Graphics::Collada::Image	Image class.
sce::SampleUtil::Graphics::Collada::InstanceMeshPart	Instance of a mesh part.
sce::SampleUtil::Graphics::Collada::InstanceMesh	Instance of a mesh.
sce::SampleUtil::Graphics::Collada::Node	Node of a scene.
sce::SampleUtil::Graphics::Collada::Animation	Animation.
sce::SampleUtil::Graphics::Collada::VisualScene	Scene class.
sce::SampleUtil::Graphics::Collada::InstanceVisualScene	Instance of a scene.
sce::SampleUtil::Graphics::Collada::AnimationPlayer	Animation player.

SCE CONFIDENTIAL

Item	Description
sce::SampleUtil::Graphics::BlendInfo	The description about blending and masking to the color surface of a shader result.
sce::SampleUtil::Graphics::GraphicsContextOption	Structure for initializing GraphicsContext.
sce::SampleUtil::Graphics::GraphicsContext	Graphics context.
sce::SampleUtil::Graphics::Effect	Name space associated with the effect.
sce::SampleUtil::Graphics::Effect::ParameterValue	The value of a shader parameter.
sce::SampleUtil::Graphics::Effect::EffectParameter	Uniform parameter with a value in an effect.
sce::SampleUtil::Graphics::Effect::EffectShader	The shader information in an effect.
sce::SampleUtil::Graphics::Effect::EffectData	Effect class.
sce::SampleUtil::Graphics::Effect::EffectInstance	Instance of an effect.
sce::SampleUtil::Graphics::FontParam	Structure used to initialize the Font class.
sce::SampleUtil::Graphics::Font	Class to handle the font texture.
sce::SampleUtil::Graphics::FontLoader	Class that generates fonts.
sce::SampleUtil::Graphics::VertexProgram	Vertex program.
sce::SampleUtil::Graphics::FragmentProgram	Fragment program.
sce::SampleUtil::Graphics::VertexStream	Vertex stream.
sce::SampleUtil::Graphics::VertexAttrib	Structure to indicate the layout of VertexBuffer.
sce::SampleUtil::Graphics::GraphicsLoader	Class to manage the resources required for drawing of graphics.
sce::SampleUtil::Graphics::Gxm::CommonDialogUpdateParam	Parameter for sceCommonDialogUpdate.
sce::SampleUtil::Graphics::Parameter	Parameter class of a shader.
sce::SampleUtil::Graphics::Program	ID of a shader program.
sce::SampleUtil::Graphics::SpriteRenderer	2D sprite rendering class
sce::SampleUtil::Graphics::Object3dRenderer	3D object renderer class
sce::SampleUtil::Input::PadContextOption	Structure for initializing PadContext.
sce::SampleUtil::Input::TouchPadData	Touch point data.
sce::SampleUtil::Input::MagnetometerData	Magnetometer data.
sce::SampleUtil::Input::MotionData	Motion data.
sce::SampleUtil::Input::PadData	Output data received from the controller.
sce::SampleUtil::Input::PadContext	Class for handling controller operation.
sce::SampleUtil::Input	Input-associated definitions.
sce::SampleUtil::Input::ControllerContextOption	Structure for initializing ControllerContext.
sce::SampleUtil::Input::ControllerContext	Class for handling controller operation.
sce::SampleUtil::Input::MotionContextData	Structure that handles device states and motion sensor values.
sce::SampleUtil::Input::MotionContext	Class that handles motion sensor data.

SCE CONFIDENTIAL

Item	Description
sce::SampleUtil::Input::TouchContextOption	Structure for setting TouchContext.
sce::SampleUtil::Input::TouchPrimitiveGestureEvent	Structure for getting the primitive gesture event.
sce::SampleUtil::Input::TouchGestureEvent	Structure for getting the gesture event.
sce::SampleUtil::Input::TouchContext	Class to handle input data from the touch panel.
sce::SampleUtil::Resource	Resource base class.
sce::SampleUtil::SampleSkeleton	Sample skeleton.
sce::SampleUtil::SampleSkeleton::SampleSkeletonOption	Sample skeleton option.
sce::SampleUtil::System	System-associated definitions.
sce::SampleUtil::System::UserIdList	Structure that stores the user ID list.
sce::SampleUtil::System::UserLoginStatusChangedEvents	Structure that stores the user login status (logged in/logged out) change events.
sce::SampleUtil::System::UserIdManager	Class for managing user IDs.
sce::SampleUtil::System::UserIdManagerOption	Options structure for UserIdManager.
sce::SampleUtil::System::SaveDataSlotParam	Save data slot parameter structure.
sce::SampleUtil::System::SaveDataSearchCond	Save data slot search condition structure.
sce::SampleUtil::System::SaveDataSearchResult	Save data slot search result structure.
sce::SampleUtil::System::DataSaveItem	Structure for specifying target save data to save.
sce::SampleUtil::System::SaveDataMountPoint	Save data mount point structure.
sce::SampleUtil::System::DataRemoveItem	Structure for specifying target save data to remove.
sce::SampleUtil::System::SaveData	Save data.
sce::SampleUtil::System::SaveDataOption	Structure for initializing SaveData.

SCE CONFIDENTIAL

sce::SampleUtil

000004892117

Summary

sce::SampleUtil

Sample utility.

Definition

```
namespace SampleUtil { }
```

Description

These are the sample utility definitions.

Function Summary

Function	Description
destroy	Discard resources.

Inner Classes, Structures, and Namespaces

Item	Description
sce::SampleUtil::Audio	Audio-associated definitions.
sce::SampleUtil::Debug	Debug-associated definitions.
sce::SampleUtil::Graphics	Graphics-associated definitions.
sce::SampleUtil::Input	Input-associated definitions.
sce::SampleUtil::Resource	Resource base class.
sce::SampleUtil::SampleSkeleton	Sample skeleton.
sce::SampleUtil::System	System-associated definitions.

Functions

destroy

Discard resources.

Definition

```
#include <sampleutil_common.h>
namespace sce {
    namespace SampleUtil {
        int destroy(
            Resource *resource
        );
    }
}
```

Arguments

resource [Resource](#) object

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This discards an instance to free resources.

SCE CONFIDENTIAL

sce::SampleUtil::Audio

000004892117

Summary

sce::SampleUtil::Audio

Audio-associated definitions.

Definition

```
namespace Audio {}
```

Description

These are the audio-associated definitions.

Variables

Public Variables

```
static const int LOOP_NUM_INFINITE Infinite loop.
```

Function Summary

Function	Description
createAudioContext	<code>AudioContext</code> generation.
createVoiceDataFromFile	Generate voice data from a file.
createVoiceDataFromGeneratorSetting	Generate voice data as a signal generator.

Inner Classes, Structures, and Namespaces

Item	Description
sce::SampleUtil::Audio::AudioContextOption	<code>AudioContext</code> option.
sce::SampleUtil::Audio::Buss	<code>Buss</code> class.
sce::SampleUtil::Audio::AudioContext	Context class for playing a sound.
sce::SampleUtil::Audio::GeneratorSettings	Parameter used for the initialization of signal generator format voice data.
sce::SampleUtil::Audio::VoiceData	<code>Voice</code> data type.
sce::SampleUtil::Audio::Envelope	Structure used to set an envelope.
sce::SampleUtil::Audio::Volume	Structure used to set a volume value.
sce::SampleUtil::Audio::Voice	Class to provide the voice playing back function.

Enumerated Types

VoiceState

The state of a voice.

Definition

```
#include <voice.h>
namespace sce {
    namespace SampleUtil {
        namespace Audio {
            enum VoiceState {
                kVoiceStateAvailable,
                kVoiceStatePause,
                kVoiceStateKeyoff,
                kVoiceStatePlaying,
                kVoiceStateInvalid
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kVoiceStateAvailable	N/A	Usable conditions.
kVoiceStatePause	N/A	Paused.
kVoiceStateKeyoff	N/A	Keying off.
kVoiceStatePlaying	N/A	Playing.
kVoiceStateInvalid	N/A	Invalid condition.

Description

This gets the state of a voice.

Type Definitions

AudioContextUpdate

Function called upon audio data update.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Audio {
            typedef void (*AudioContextUpdate)((
                void *mainBuffer,
                uint32_t mainBufferSize,
                void *bgmBuffer,
                uint32_t bgmBufferSize,
                void *userArg
            ));
        }
    }
}
```

Arguments

<i>mainBuffer</i>	This is the data sent to the main port of the audio driver. SCE_AUDIO_OUT_PARAM_FORMAT_S16_STEREO format
<i>mainBufferSize</i>	<i>mainBuffer</i> size (bytes)
<i>bgmBuffer</i>	This is the data sent to the BGM port of the audio driver. SCE_AUDIO_OUT_PARAM_FORMAT_S16_STEREO format
<i>bgmBufferSize</i>	<i>bgmBuffer</i> size (bytes)
<i>userArg</i>	Value specified upon AudioContext initialization

Return Values

None

Description

This function is called immediately before the data generated by [AudioContext](#) is sent to the audio driver.

Functions

createAudioContext

[AudioContext](#) generation.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Audio {
            int createAudioContext(
                AudioContext **outAudioContext,
                const AudioContextOption *option = NULL
            );
        }
    }
}
```

Arguments

<i>outAudioContext</i>	Pointer to which the generated AudioContext returns
<i>option</i>	Pointer to AudioContextOption (Omissible)

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid
SCE_SAMPLE_UTIL_ERROR_FATAL	Initialization failure
<SCE_OK	Failure (Ngs2 library error code)

Description

This generates an [AudioContext](#) class. It is possible to specify the callback function used when the audio output is updated with the argument contextUpdate. To delete the generated instance, use [sce::SampleUtil::destroy\(\)](#).

SCE CONFIDENTIAL

createVoiceDataFromFile

Generate voice data from a file.

Definition

```
#include <data.h>
namespace sce {
    namespace SampleUtil {
        namespace Audio {
            int createVoiceDataFromFile(
                VoiceData **outVoiceData,
                const char *path,
                bool isStreaming
            );
        }
    }
}
```

Arguments

<i>outVoiceData</i>	Pointer to which the generated voice data returns
<i>path</i>	Path of a data file
<i>isStreaming</i>	Set whether to initialize as streaming.

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Failure

Description

This generates voice data from a file. If *isStreaming* is true, streaming is used to read the data from the file. If *isStreaming* is false, all the data in the memory is read when voice data is generated.

SCE CONFIDENTIAL

createVoiceDataFromGeneratorSetting

Generate voice data as a signal generator.

Definition

```
#include <data.h>
namespace sce {
    namespace SampleUtil {
        namespace Audio {
            int createVoiceDataFromGeneratorSetting(
                VoiceData **outVoiceData,
                const GeneratorSettings *setting
            );
        }
    }
}
```

Arguments

<i>outVoiceData</i>	Pointer to which the generated voice data returns
<i>setting</i>	Setting of signal generator

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Failure

Description

Generates voice data as a signal generator.

SCE CONFIDENTIAL

sce::SampleUtil::Audio::AudioContextOption

Summary

sce::SampleUtil::Audio::AudioContextOption

[AudioContext](#) option.

Definition

```
#include <context.h>
struct AudioContextOption {};
```

Description

This is the [AudioContext](#) option.

Fields

Public Instance Fields

AudioContextUpdate	Function called immediately before AudioContext sends data to the driver.
contextUpdate	
void *userArg	User-specified argument of contextUpdate.
uint32_t stereoVoiceNum	Stereo voice maximum value (Default value: 8)
uint32_t monoVoiceNum	Monaural voice maximum value (Default value: 32)
uint32_t at9VoiceNum	Maximum value for voice using ATRAC9™ (Default value: 4)
uint32_t reverbBussNum	Reverb buss maximum value (Default value: 4)
uint32_t mixerBussNum	Mixer buss maximum value (Default value: 4)

SCE CONFIDENTIAL

sce::SampleUtil::Audio::Buss

000004892117

Summary

sce::SampleUtil::Audio::Buss

Buss class.

Definition

```
#include <context.h>
class Buss : public sce::SampleUtil::Resource {};
```

Description

This is the buss class. This is used for audio routing.

Methods Summary

Methods	Description
setOutput	Output destination setting.

Public Instance Methods

setOutput

Output destination setting.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Audio {
            class Buss {
                virtual int setOutput(
                    uint32_t nSourceOutputIndex,
                    Buss *buss
                )=0;
            }
        }
    }
}
```

Arguments

<i>nSourceOutputIndex</i> <i>buss</i>	Output index. 0 or higher and lower than 4. This is the buss of the output destination. If specified as NULL, there will be no output anywhere.
--	--

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This sets the output destination for audio data. By default the index 0 is connected to the main buss. If the buss is specified as NULL, there will be no output anywhere.

sce::SampleUtil::Audio::AudioContext

000004892117

Summary

sce::SampleUtil::Audio::AudioContext

Context class for playing a sound.

Definition

```
#include <context.h>
class AudioContext : public sce::SampleUtil::Resource {};
```

Description

This is the class to handle resources for playing sound. [AudioContext](#) is specified by initialization of [Voice](#) and ReverbBuss after executing initialize for itself. It is possible to use one [AudioContext](#) instance for multiple Voices and ReverbBusses.

Methods Summary

Methods	Description
createReverbBuss	Creates a reverb buss.
createVoice	Create a voice.
getMainBuss	Return the main buss.
getBgmBuss	Return the BGM buss.

Public Instance Methods

createReverbBuss

Creates a reverb buss.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Audio {
            class AudioContext {
                virtual int createReverbBuss(
                    Buss **outBuss,
                    const int reverbMode
                )=0;
            }
        }
    }
}
```

Arguments

<i>outBuss</i>	Pointer to which the generated buss returns
<i>reverbMode</i>	Reverberation mode

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid
SCE_SAMPLE_UTIL_ERROR_BUSY	No usable reverb voice exists
<SCE_OK	Failure

Description

This creates a reverb buss.

SCE CONFIDENTIAL

createVoice

Create a voice.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Audio {
            class AudioContext {
                virtual int createVoice(
                    Voice **outVoice,
                    const VoiceData *voiceData
                )=0;
            }
        }
    }
}
```

Arguments

outVoice Pointer to which the generated voice returns
voiceData [Voice](#) data

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This generates a voice with the voice data as the source. To delete the generated instance, use [sce::SampleUtil::destroy\(\)](#).

SCE CONFIDENTIAL

getMainBuss

Return the main buss.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Audio {
            class AudioContext {
                virtual Buss *getMainBuss(void)=0;
            }
        }
    }
}
```

Return Values

Main buss

Description

This returns the buss connected to the main port of the audio driver.

SCE CONFIDENTIAL

getBgmBuss

Return the BGM buss.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Audio {
            class AudioContext {
                virtual Buss *getBgmBuss(void)=0;
            }
        }
    }
}
```

Return Values

BGM buss

Description

This returns the buss connected to the BGM port of the audio driver.

SCE CONFIDENTIAL

sce::SampleUtil::Audio::GeneratorSettings

Summary

sce::SampleUtil::Audio::GeneratorSettings

Parameter used for the initialization of signal generator format voice data.

Definition

```
#include <data.h>
struct GeneratorSettings {};
```

Description

This is the parameter used for the initialization of signal generator format voice data.

Fields

Public Instance Fields

uint32_t <i>generatorMode</i>	This is the type of wave forms generated. Specify NGS wave forms such as SCE_NGS_GENERATOR_SINE.
int32_t <i>frequency</i>	Frequency response (Hz) of wave forms.
float <i>amplitude</i>	This is the amplitude of wave forms. 1 = complete amplitude (no clipping), 0.5 = half amplitude.
float <i>pulseWidth</i>	Pulse width (upon pulse wave form selection)
uint32_t <i>sampleOffset</i>	Generation start offset within wave form (for example, ability to generate sine waves as cosine waves)
uint32_t <i>phaseAngle</i>	Start phase offset at the time of sine wave generation (designate as angle (the degree of) between 0 and 360)

SCE CONFIDENTIAL

sce::SampleUtil::Audio::VoiceData

000004892117

Summary

sce::SampleUtil::Audio::VoiceData

Voice data type.

Definition

```
#include <data.h>
class VoiceData : public sce::SampleUtil::Resource {};
```

Description

This is the voice data type for generating the voice instance.

SCE CONFIDENTIAL

sce::SampleUtil::Audio::Envelope

000004892117

Summary

sce::SampleUtil::Audio::Envelope

Structure used to set an envelope.

Definition

```
#include <voice.h>
struct Envelope {};
```

Description

For details on envelopes, refer to Chapter 6 "Amplitude Envelope DSP Effect Module Overview" in the "NGS Modules Overview" document.

Fields

Public Instance Fields

SceNgsEnvelopePoint <i>envelopePoints</i>	Structure required for initialization of the NGS envelope DSP effect module.
uint32_t <i>releaseMsecs</i>	Release rate (msec)
uint32_t <i>numPoints</i>	Number of points in the envelope (1-4)
uint32_t <i>loopStart</i>	Loop start number of the envelope (0-2)
int32_t <i>loopEnd</i>	Loop end number of the envelope (1-3, must be larger than loopStart)

SCE CONFIDENTIAL

sce::SampleUtil::Audio::Volume

000004892117

Summary

sce::SampleUtil::Audio::Volume

Structure used to set a volume value.

Definition

```
#include <voice.h>
struct Volume {};
```

Description

When 1.0f or 0.5f is specified, the volume is adjusted to the same volume as the original volume or half the original volume respectively.

Fields

Public Instance Fields

float *left*
float *right*

[Volume](#) value for the left channel.
[Volume](#) value for the right channel.

SCE CONFIDENTIAL

sce::SampleUtil::Audio::Voice

000004892117

Summary

sce::SampleUtil::Audio::Voice

Class to provide the voice playing back function.

Definition

```
#include <voice.h>
class Voice : public sce::SampleUtil::Resource {};
```

Description

This is the class for an application to easily play sound. It is necessary to initialize the [AudioContext](#) class before executing initialize for the [Voice](#) class.

Methods Summary

Methods	Description
play	Plays a voice.
keyOff	Key-off of a voice.
kill	Stop voice playback.
pause	Pauses the voice being played.
resume	Resumes voice playback.
setEnvelope	Sets the voice envelope information.
setOutput	Connects a voice and a buss.
getState	Gets the state of a voice.
setVolume	Volume settings.

Public Instance Methods

play

Plays a voice.

Definition

```
#include <voice.h>
namespace sce {
    namespace SampleUtil {
        namespace Audio {
            class Voice {
                virtual int play(
                    int loopCount = 0
                )=0;
            }
        }
    }
}
```

Arguments

loopCount Loop count

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Invalid initialization parameter (VoiceInitParam)
<SCE_OK	Failure

Description

This plays voice.

SCE CONFIDENTIAL

keyOff

Key-off of a voice.

Definition

```
#include <voice.h>
namespace sce {
    namespace SampleUtil {
        namespace Audio {
            class Voice {
                virtual int keyOff(void)=0;
            }
        }
    }
}
```

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Invalid initialization parameter (VoiceInitParam)
<SCE_OK	Failure

Description

Performs a key-off of a voice.

SCE CONFIDENTIAL

kill

Stop voice playback.

Definition

```
#include <voice.h>
namespace sce {
    namespace SampleUtil {
        namespace Audio {
            class Voice {
                virtual int kill(void)=0;
            }
        }
    }
}
```

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Invalid initialization parameter (VoiceInitParam)
<SCE_OK	Failure

Description

This stops voice play.

SCE CONFIDENTIAL

pause

Pauses the voice being played.

Definition

```
#include <voice.h>
namespace sce {
    namespace SampleUtil {
        namespace Audio {
            class Voice {
                virtual int pause(void)=0;
            }
        }
    }
}
```

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Invalid initialization parameter (VoiceInitParam)
<SCE_OK	Failure

Description

Pauses the voice being played.

SCE CONFIDENTIAL

resume

Resumes voice playback.

Definition

```
#include <voice.h>
namespace sce {
    namespace SampleUtil {
        namespace Audio {
            class Voice {
                virtual int resume(void)=0;
            }
        }
    }
}
```

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Invalid initialization parameter (VoiceInitParam)
<SCE_OK	Failure

Description

Restarts playing the voice paused.

SCE CONFIDENTIAL

setEnvelope

Sets the voice envelope information.

Definition

```
#include <voice.h>
namespace sce {
    namespace SampleUtil {
        namespace Audio {
            class Voice {
                virtual int setEnvelope(
                    const Envelope *envelopeInfo
                )=0;
            }
        }
    }
}
```

Arguments

envelopeInfo Pointer to the envelop information structure

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid
<SCE_OK	Failure

Description

Sets the voice envelope information.

SCE CONFIDENTIAL

setOutput

Connects a voice and a buss.

Definition

```
#include <voice.h>
namespace sce {
    namespace SampleUtil {
        namespace Audio {
            class Voice {
                virtual int setOutput(
                    uint32_t outIndex,
                    Buss *buss
                )=0;
            }
        }
    }
}
```

Arguments

<i>outIndex</i>	The voice output index
<i>buss</i>	The connecting buss

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid
SCE_SAMPLE_UTIL_ERROR_FATAL	Buss connection failure
<SCE_OK	Failure

Description

Connects the output destination of a voice with a buss.

SCE CONFIDENTIAL

getState

Gets the state of a voice.

Definition

```
#include <voice.h>
namespace sce {
    namespace SampleUtil {
        namespace Audio {
            class Voice {
                virtual VoiceState getState(void)=0 const;
            }
        }
    }
}
```

Return Values

Value	Description
<code>>=SCE_OK</code>	The state of the voice (Normal termination)
<code>SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM</code>	Invalid initialization parameter (VoiceInitParam)
<code><SCE_OK</code>	Failure

Description

Gets the state of a voice (VoiceState).

SCE CONFIDENTIAL

setVolume

[Volume](#) settings.

Definition

```
#include <voice.h>
namespace sce {
    namespace SampleUtil {
        namespace Audio {
            class Voice {
                virtual int setVolume(
                    uint32_t outIndex,
                    const Volume *volume
                )=0;
            }
        }
    }
}
```

Arguments

outIndex The output destination of a voice
volume [Volume](#)

Return Values

Value	Description
<code>>=SCE_OK</code>	The state of the voice (Normal termination)
<code><SCE_OK</code>	Failure

Description

This sets the volume.

sce::SampleUtil::Debug

000004892117

Summary

sce::SampleUtil::Debug

Debug-associated definitions.

Definition

```
namespace Debug {}
```

Description

These are the Debug-associated definitions.

Function Summary

Function	Description
createConsole	Create Console instance.
createMenu	Create empty menu class.
createMenuFromXmlString	Create menu class from Xml.

Inner Classes, Structures, and Namespaces

Item	Description
sce::SampleUtil::Debug::Console	Class that handles the debug console.
sce::SampleUtil::Debug::AbstractMenuItem	Base class of each menu item class.
sce::SampleUtil::Debug::Int8MenuItem	Debug menu item class that can manage signed 8-bit integer type values.
sce::SampleUtil::Debug::Int16MenuItem	Debug menu item class that can manage signed 16-bit integer type values.
sce::SampleUtil::Debug::Int32MenuItem	Debug menu item class that can manage signed 32-bit integer type values.
sce::SampleUtil::Debug::Int64MenuItem	Debug menu item class that can manage signed 64-bit integer type values.
sce::SampleUtil::Debug::Uint8MenuItem	Debug menu item class that can manage unsigned 8-bit integer type values.
sce::SampleUtil::Debug::Uint16MenuItem	Debug menu item class that can manage unsigned 16-bit integer type values.
sce::SampleUtil::Debug::Uint32MenuItem	Debug menu item class that can manage unsigned 32-bit integer type values.
sce::SampleUtil::Debug::Uint64MenuItem	Debug menu item class that can manage unsigned 64-bit integer type values.
sce::SampleUtil::Debug::FloatMenuItem	Debug menu item class that can manage single precision floating-point type values.
sce::SampleUtil::Debug::DoubleMenuItem	Debug menu item class that can manage double precision floating-point type values.
sce::SampleUtil::Debug::BoolMenuItem	Debug menu item class that can manage Bool type values.
sce::SampleUtil::Debug::EnumMenuItem	Debug menu item class that can manage indexed character string set types.
sce::SampleUtil::Debug::MenuItemGroup	Debug menu item class that can group multiple AbstractMenuItems.

SCE CONFIDENTIAL

Item	Description
<u>sce::SampleUtil::Debug::Menu</u>	Class that performs management/rendering/etc. for a menu that is a tree structure.

000004892117

Enumerated Types

Type

Constant that indicates the menu item type.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            enum Type {
                TYPE_GROUP,
                TYPE_FLOAT,
                TYPE_DOUBLE,
                TYPE_INT8,
                TYPE_INT16,
                TYPE_INT32,
                TYPE_INT64,
                TYPE_UINT8,
                TYPE_UINT16,
                TYPE_UINT32,
                TYPE_UINT64,
                TYPE_BOOL,
                TYPE_ENUM,
                TYPE_INT = TYPE_INT32
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
TYPE_GROUP	N/A	Constant that indicates a container type menu item.
TYPE_FLOAT	N/A	Constant that indicates a single precision floating-point type menu item.
TYPE_DOUBLE	N/A	Constant that indicates a double precision floating-point type menu item.
TYPE_INT8	N/A	Constant that indicates a signed 8-bit integer type menu item.
TYPE_INT16	N/A	Constant that indicates a signed 16-bit integer type menu item.
TYPE_INT32	N/A	Constant that indicates a signed 32-bit integer type menu item.
TYPE_INT64	N/A	Constant that indicates a signed 64-bit integer type menu item.
TYPE_UINT8	N/A	Constant that indicates an unsigned 8-bit integer type menu item.
TYPE_UINT16	N/A	Constant that indicates an unsigned 16-bit integer type menu item.
TYPE_UINT32	N/A	Constant that indicates an unsigned 32-bit integer type menu item.

SCE CONFIDENTIAL

Macro	Value	Description
TYPE_UINT64	N/A	Constant that indicates an unsigned 64-bit integer type menu item.
TYPE_BOOL	N/A	Constant that indicates a Bool type menu item.
TYPE_ENUM	N/A	Constant that indicates an indexed character string set type menu item.
TYPE_INT	TYPE_INT32	Constant that indicates an unsigned 32-bit integer type menu item.

Description

This constant indicates the menu item type. Each menu item type can be identified using the [`sce::SampleUtil::Debug::AbstractMenuItem::getType`](#) method.

Type Definitions

Callback

Definition of callback function for menu items.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            typedef void (*Callback)(  

                AbstractMenuItem *item,  

                void *callbackArg,  

                Input::Button pressedButtons  

            );
        }
    }
}
```

Arguments

<i>item</i>	Pointer to AbstractMenuItem
<i>callbackArg</i>	Pointer to the argument set when registering a callback
<i>pressedButtons</i>	Button type pressed that caused the callback to occur

Return Values

None

Description

This is the definition of the function that can be registered (using the [sce::SampleUtil::Debug::AbstractMenuItem::setCallback](#) method) as the callback function called when each frame is updated.

SCE CONFIDENTIAL

OnUpdateCallback

Definition of callback function for menu items.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            typedef void (*OnUpdateCallback)(AbstractMenuItem *item,
                                              void *callbackArg,
                                              const Input::PadContext *padContext
                                             );
        }
    }
}
```

Arguments

<i>item</i>	Pointer to AbstractMenuItem
<i>callbackArg</i>	Pointer to the argument set when registering a callback
<i>padContext</i>	PadContext passed to the update function

Return Values

None

Description

This is the definition of the function that can be registered (using the [sce::SampleUtil::Debug::AbstractMenuItem::setCallback](#) method) as the callback function called when each frame is updated.

Functions

createConsole

Create [Console](#) instance.

Definition

```
#include <console.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            int createConsole(
                Console **console,
                float fontHeight,
                uint32_t lines,
                uint32_t nCharsPerLine,
                float lineSpacing = 0.0
            );
        }
    }
}
```

Arguments

<i>console</i>	Reference to the pointer for the created Console instance
<i>fontHeight</i>	Font height
<i>lines</i>	Number of lines in the console to create
<i>nCharsPerLine</i>	Number of characters per line in the console to create. When output of a character string longer than the number of characters specified here is attempted, it will be wrapped at the right edge.
<i>lineSpacing</i>	Line spacing

Return Values

Value	Description
SCE_OK	Success

Description

This creates a [Console](#) instance. To delete an instance, use [sce::SampleUtil::destroy\(\)](#).

SCE CONFIDENTIAL

createMenu

Create empty menu class.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            int createMenu(
                Menu **menu,
                const char *label
            );
        }
    }
}
```

Arguments

<i>menu</i>	Reference to a pointer which stores the generated Menu
<i>label</i>	Menu item descriptor

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Failure (error code)

Description

This creates a Class [Menu](#) instance without any menu items registered.

SCE CONFIDENTIAL

createMenuFromXmlString

Create menu class from Xml.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            int createMenuFromXmlString(
                Menu **menu,
                const char *xml
            );
        }
    }
}
```

Arguments

<i>menu</i>	Reference to a pointer which stores the generated Menu instance
<i>xml</i>	Entire XML string that describes the menu structure

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Failure (error code)

Description

This creates a Class [Menu](#) instance from xml that describes a menu structure.

sce::SampleUtil::Debug::Console

000004892117

Summary

sce::SampleUtil::Debug::Console

Class that handles the debug console.

Definition

```
#include <console.h>
class Console : public sce::SampleUtil::Resource {};
```

Description

This class handles the debug console.

Methods Summary

Methods	Description
~Console	Destructor.
draw	Debug console rendering.
vprintf	Output with formatting.
printf	Output with formatting.
setBgColor	Background color setting.
setTextColor	Text color setting.
enableTtyOutput	Enable/disable TTY output.

Constructors and Destructors

~Console

Destructor.

Definition

```
#include <console.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Console {
                virtual inline ~Console();
            };
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Public Instance Methods

draw

Debug console rendering.

Definition

```
#include <console.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Console {
                virtual int draw(
                    sce::SampleUtil::Graphics::GraphicsContext *gfxContext,
                    sce::SampleUtil::Graphics::SpriteRenderer *spriteRender,
                    sce::Vectormath::Simd::Aos::Vector2
                    position
                    )=0;
            };
        }
    }
}
```

Arguments

<i>gfxContext</i>	Pointer to GraphicsContext
<i>spriteRender</i>	Pointer to SpriteRenderer
<i>position</i>	Position of drawing (Specifies the upper left coordinate)

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_NULL_POINTER	When a NULL pointer is passed to the argument

Description

Performs debug console rendering.

SCE CONFIDENTIAL

vprintf

Output with formatting.

Definition

```
#include <console.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Console {
                virtual int vprintf(
                    const char *format,
                    va_list ap
                )=0;
            }
        }
    }
}
```

Arguments

<i>format</i>	printf format formatting
<i>ap</i>	Variable length argument

Return Values

Value	Description
(≥ 0)	Success. Number of characters output.
SCE_SAMPLE_UTIL_ERROR_FATAL	realloc failure

Description

This outputs a character string with formatting.

SCE CONFIDENTIAL

printf

Output with formatting.

Definition

```
#include <console.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Console {
                virtual int printf(
                    const char *format,
                    ...
                )=0;
            }
        }
    }
}
```

Arguments

<i>format</i>	printf format formatting
...	Variable length argument

Return Values

Value	Description
(≥ 0)	Success. Number of characters output.
SCE_SAMPLE_UTIL_ERROR_FATAL	realloc failure

Description

This outputs a character string with formatting.

SCE CONFIDENTIAL

setBgColor

Background color setting.

Definition

```
#include <console.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Console {
                virtual void setBgColor(
                    sce::Vectormath::Simd::Aos::Vector4_arg
color
                )=0;
            }
        }
    }
}
```

Arguments

color Background color

Return Values

None

Description

This Sets the background color of the console.

SCE CONFIDENTIAL

setTextColor

Text color setting.

Definition

```
#include <console.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Console {
                virtual void setTextColor(
                    sce::Vectormath::Simd::Aos::Vector4_arg
color
                )=0;
            }
        }
    }
}
```

Arguments

color Text color

Return Values

None

Description

This Sets the color of the text output to the console.

SCE CONFIDENTIAL

enableTtyOutput

Enable/disable TTY output.

Definition

```
#include <console.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Console {
                virtual void enableTtyOutput(
                    bool enable
                )=0;
            }
        }
    }
}
```

Arguments

enable Enable (true)/disable (false) character string output to the TTY

Return Values

None

Description

When enable is true, the character strings output to the console will also be output to the TTY. When false, only output to the console will be performed, not the TTY.

sce::SampleUtil::Debug::AbstractMenuItem

Summary

sce::SampleUtil::Debug::AbstractMenuItem

Base class of each menu item class.

Definition

```
#include <menu.h>
class AbstractMenuItem {};
```

Description

Base class that holds shared methods for each menu item class

Methods Summary

Methods	Description
~AbstractMenuItem	Destructor.
getType	Obtain the instance menu item type.
getId	Obtains the instance ID.
isVisible	Checks if a menu item is visible/hidden.
setVisible	Sets a menu item to visible/hidden.
isActive	Checks if operation for a menu item is active or inactive.
 setActive	Sets operation for a menu item to active/inactive.
setCallback	Callback function registration.
setOnUpdateCallback	Callback function registration.
toString	Generate character string from menu item.
update	Menu update.
update	Menu update.

Constructors and Destructors

~AbstractMenuItem

Destructor.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class AbstractMenuItem {
                virtual inline ~AbstractMenuItem();
            }
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Public Instance Methods

getType

Obtain the instance menu item type.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class AbstractMenuItem {
                virtual Type getType(void)=0 const;
            }
        }
    }
}
```

Return Values

[Menu](#) item type defined with [sce::SampleUtil::Debug::Type](#)

Description

This obtains the type of menu item for an instance.

SCE CONFIDENTIAL

getId

Obtains the instance ID.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class AbstractMenuItem {
                virtual const char *getId(void)=0 const;
            }
        }
    }
}
```

Return Values

Instance ID

Description

This obtains the ID allocated at instance creation.

SCE CONFIDENTIAL

isVisible

Checks if a menu item is visible/hidden.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class AbstractMenuItem {
                virtual bool isVisible(void)=0 const;
            }
        }
    }
}
```

Return Values

Value	Description
true	Visible
false	Hidden

Description

Checks if a menu item is visible/hidden.

SCE CONFIDENTIAL

setVisible

Sets a menu item to visible/hidden.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class AbstractMenuItem {
                virtual void setVisible(
                    bool visible
                )=0;
            }
        }
    }
}
```

Arguments

visible visible (true)/hidden (false)

Return Values

None

Description

Sets if a menu item is visible/hidden.

SCE CONFIDENTIAL

isActive

Checks if operation for a menu item is active or inactive.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class AbstractMenuItem {
                virtual bool isActive(void)=0 const;
            }
        }
    }
}
```

Return Values

Value	Description
true	Operation for the menu is active
false	Operation for the menu is inactive

Description

Checks if button operation for a menu item is active or inactive. If button operation is active, the value can be increased/decreased with the left/right buttons. The value increased/decreased with this operation will also be reflected in bound variables.

SCE CONFIDENTIAL

setActive

Sets operation for a menu item to active/inactive.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class AbstractMenuItem {
                virtual void setActive(
                    bool active
                )=0;
            }
        }
    }
}
```

Arguments

active Active (true)/inactive (false)

Return Values

None

Description

Switches operation for a menu item to active/inactive

SCE CONFIDENTIAL

setCallback

Callback function registration.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class AbstractMenuItem {
                virtual void setCallback(
                    Callback callback,
                    void *callbackArg
                )=0;
            }
        }
    }
}
```

Arguments

<i>callback</i>	Pointer to the callback function
<i>callbackArg</i>	Memory area address to pass to the callback function as an argument

Return Values

None

Description

This registers the function that calls back the button operation for menu items during updates for each frame.

SCE CONFIDENTIAL

setOnUpdateCallback

Callback function registration.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class AbstractMenuItem {
                virtual void setOnUpdateCallback(
                    OnUpdateCallback onUpdateCallback,
                    void *callbackArg
                )=0;
            }
        }
    }
}
```

Arguments

<i>onUpdateCallback</i>	Pointer to the callback function
<i>callbackArg</i>	Pointer to pass to the callback function as an argument

Return Values

None

Description

Registers the function to be called when the cursor is placed over this menu for update of each frame.

SCE CONFIDENTIAL

toString

Generate character string from menu item.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class AbstractMenuItem {
                virtual const char *toString(void)=0 const;
            }
        }
    }
}
```

Return Values

Generated character string

Description

This generates a character string that reflects the menu item content.

SCE CONFIDENTIAL

update

Menu update.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class AbstractMenuItem {
                virtual void update(
                    const Input::ControllerContext *ctrl
                )=0;
            }
        }
    }
}
```

Arguments

ctrl Pointer to PadContext

Return Values

None

Description

Updates the menu status.

SCE CONFIDENTIAL

update

Menu update.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class AbstractMenuItem {
                virtual void update(
                    const Input::PadContext *pad
                )=0;
            }
        }
    }
}
```

Arguments

pad Pointer to PadContext

Return Values

None

Description

Updates the menu status.

sce::SampleUtil::Debug::Int8MenuItem

Summary

sce::SampleUtil::Debug::Int8MenuItem

Debug menu item class that can manage signed 8-bit integer type values.

Definition

```
#include <menu.h>
class Int8MenuItem : public sce::SampleUtil::Debug::AbstractMenuItem {};
```

Description

This is the debug menu item class that can manage (set/obtain/bind) signed 8-bit integer type values.

Methods Summary

Methods	Description
~Int8MenuItem	Destructor.
getValue	Get signed 8-bit integer type value that is currently set.
setValue	Set signed 8-bit integer type value.
setReferenceValue	Bind signed 8-bit integer type variable to menu item.
downcast	Downcast function for Int8MenuItem .

Constructors and Destructors

~Int8MenuItem

Destructor.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Int8MenuItem {
                virtual inline ~Int8MenuItem();
            };
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Static Methods

downcast

Downcast function for [Int8MenuItem](#).

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Int8MenuItem {
                static Int8MenuItem *downcast(
                    AbstractMenuItem *menuItem
                );
            };
        }
    }
}
```

Arguments

menuItem Pointer to [AbstractMenuItem](#)

Return Values

Value	Description
Int8MenuItem*	If the downcast is successful, a pointer to Int8MenuItem will be returned
NULL	If downcast failed

Description

This downcasts an [AbstractMenuItem](#) pointer to [Int8MenuItem](#).

Public Instance Methods

getValue

Get signed 8-bit integer type value that is currently set.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Int8MenuItem {
                virtual int8_t getValue(void)=0 const;
            }
        }
    }
}
```

Return Values

Current signed 8-bit integer type value

Description

This obtains the signed 8-bit integer type value that is currently set.

SCE CONFIDENTIAL

setValue

Set signed 8-bit integer type value.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Int8MenuItem {
                virtual void setValue(
                    int8_t value
                )=0;
            }
        }
    }
}
```

Arguments

value Signed 8-bit integer type value

Return Values

None

Description

This sets the signed 8-bit integer type value.

SCE CONFIDENTIAL

setReferenceValue

Bind signed 8-bit integer type variable to menu item.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Int8MenuItem {
                virtual void setReferenceValue(
                    int8_t *refValue
                )=0;
            }
        }
    }
}
```

Arguments

refValue Signed 8-bit integer type pointer

Return Values

None

Description

This binds a signed 8-bit integer type variable to a menu item.

sce::SampleUtil::Debug::Int16MenuItem

000004892117

Summary

sce::SampleUtil::Debug::Int16MenuItem

Debug menu item class that can manage signed 16-bit integer type values.

Definition

```
#include <menu.h>
class Int16MenuItem : public sce::SampleUtil::Debug::AbstractMenuItem {};
```

Description

This is the debug menu item class that can manage (set/obtain/bind) signed 16-bit integer type values.

Methods Summary

Methods	Description
~Int16MenuItem	Destructor.
getValue	Get signed 16-bit integer type value that is currently set.
setValue	Set signed 16-bit integer type value.
setReferenceValue	Bind signed 16-bit integer type variable to menu item.
downcast	Downcast function for Int16MenuItem .

Constructors and Destructors

~Int16MenuItem

Destructor.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Int16MenuItem {
                virtual inline ~Int16MenuItem();
            };
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Static Methods

downcast

Downcast function for [Int16MenuItem](#).

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Int16MenuItem {
                static Int16MenuItem *downcast(
                    AbstractMenuItem *menuItem
                );
            };
        }
    }
}
```

Arguments

menuItem Pointer to [AbstractMenuItem](#)

Return Values

Value	Description
Int16MenuItem*	If downcast succeeded
NULL	If downcast failed

Description

This downcasts an [AbstractMenuItem](#) pointer to [Int16MenuItem](#).

Public Instance Methods

getValue

Get signed 16-bit integer type value that is currently set.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Int16MenuItem {
                virtual int16_t getValue(void)=0 const;
            }
        }
    }
}
```

Return Values

Current signed 16-bit integer type value

Description

This obtains the signed 16-bit integer type value that is currently set.

SCE CONFIDENTIAL

setValue

Set signed 16-bit integer type value.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Int16MenuItem {
                virtual void setValue(
                    int16_t value
                )=0;
            }
        }
    }
}
```

Arguments

value Signed 16-bit integer type value

Return Values

None

Description

This sets the signed 16-bit integer type value.

SCE CONFIDENTIAL

setReferenceValue

Bind signed 16-bit integer type variable to menu item.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Int16MenuItem {
                virtual void setReferenceValue(
                    int16_t *refValue
                )=0;
            }
        }
    }
}
```

Arguments

refValue Signed 16-bit integer type pointer

Return Values

None

Description

This binds a signed 16-bit integer type variable to a menu item.

sce::SampleUtil::Debug::Int32MenuItem

000004892117

Summary

sce::SampleUtil::Debug::Int32MenuItem

Debug menu item class that can manage signed 32-bit integer type values.

Definition

```
#include <menu.h>
class Int32MenuItem : public sce::SampleUtil::Debug::AbstractMenuItem {};
```

Description

This is the debug menu item class that can manage (set/obtain/bind) signed 32-bit integer type values.

Methods Summary

Methods	Description
~Int32MenuItem	Destructor.
getValue	Get signed 32-bit integer type value that is currently set.
setValue	Set signed 32-bit integer type value.
setReferenceValue	Bind signed 32-bit integer type variable to menu item.
downcast	Downcast function for Int32MenuItem .

Constructors and Destructors

~Int32MenuItem

Destructor.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Int32MenuItem {
                virtual inline ~Int32MenuItem();
            }
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Static Methods

downcast

Downcast function for [Int32MenuItem](#).

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Int32MenuItem {
                static Int32MenuItem *downcast(
                    AbstractMenuItem *menuItem
                );
            };
        }
    }
}
```

Arguments

menuItem Pointer to [AbstractMenuItem](#)

Return Values

Value	Description
Int32MenuItem*	If the downcast is successful, a pointer to Int32MenuItem will be returned
NULL	If downcast failed

Description

This downcasts an [AbstractMenuItem](#) pointer to [Int32MenuItem](#).

Public Instance Methods

getValue

Get signed 32-bit integer type value that is currently set.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Int32MenuItem {
                virtual int32_t getValue(void)=0 const;
            }
        }
    }
}
```

Return Values

Current signed 32-bit integer type value

Description

This obtains the signed 32-bit integer type value that is currently set.

SCE CONFIDENTIAL

setValue

Set signed 32-bit integer type value.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Int32MenuItem {
                virtual void setValue(
                    int32_t value
                )=0;
            }
        }
    }
}
```

Arguments

value Signed 32-bit integer type value

Return Values

None

Description

This sets the signed 32-bit integer type value.

SCE CONFIDENTIAL

setReferenceValue

Bind signed 32-bit integer type variable to menu item.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Int32MenuItem {
                virtual void setReferenceValue(
                    int32_t *refValue
                )=0;
            }
        }
    }
}
```

Arguments

refValue Signed 32-bit integer type pointer

Return Values

None

Description

This binds a signed 32-bit integer type variable to a menu item.

sce::SampleUtil::Debug::Int64MenuItem

000004892117

Summary

sce::SampleUtil::Debug::Int64MenuItem

Debug menu item class that can manage signed 64-bit integer type values.

Definition

```
#include <menu.h>
class Int64MenuItem : public sce::SampleUtil::Debug::AbstractMenuItem {};
```

Description

This is the debug menu item class that can manage (set/obtain/bind) signed 64-bit integer type values.

Methods Summary

Methods	Description
~Int64MenuItem	Destructor.
getValue	Get signed 64-bit integer type value that is currently set.
setValue	Set signed 64-bit integer type value.
setReferenceValue	Bind signed 64-bit integer type variable to menu item.
downcast	Downcast function for Int64MenuItem .

Constructors and Destructors

~Int64MenuItem

Destructor.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Int64MenuItem {
                virtual inline ~Int64MenuItem();
            }
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Static Methods

downcast

Downcast function for [Int64MenuItem](#).

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Int64MenuItem {
                static Int64MenuItem *downcast(
                    AbstractMenuItem *menuItem
                );
            };
        }
    }
}
```

Arguments

menuItem Pointer to [AbstractMenuItem](#)

Return Values

Value	Description
Int64MenuItem*	If the downcast is successful, a pointer to Int64MenuItem will be returned
NULL	If downcast failed

Description

This downcasts an [AbstractMenuItem](#) pointer to [Int64MenuItem](#).

Public Instance Methods

getValue

Get signed 64-bit integer type value that is currently set.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Int64MenuItem {
                virtual int64_t getValue(void)=0 const;
            }
        }
    }
}
```

Return Values

Current signed 64-bit integer type value

Description

This obtains the signed 64-bit integer type value that is currently set.

SCE CONFIDENTIAL

setValue

Set signed 64-bit integer type value.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Int64MenuItem {
                virtual void setValue(
                    int64_t value
                )=0;
            }
        }
    }
}
```

Arguments

value Signed 64-bit integer type value

Return Values

None

Description

This sets the signed 64-bit integer type value.

SCE CONFIDENTIAL

setReferenceValue

Bind signed 64-bit integer type variable to menu item.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Int64MenuItem {
                virtual void setReferenceValue(
                    int64_t *refValue
                )=0;
            }
        }
    }
}
```

Arguments

refValue Signed 64-bit integer type pointer

Return Values

None

Description

This binds a signed 64-bit integer type variable to a menu item.

sce::SampleUtil::Debug::UInt8MenuItem

Summary

sce::SampleUtil::Debug::Uint8MenuItem

Debug menu item class that can manage unsigned 8-bit integer type values.

Definition

```
#include <menu.h>
class Uint8MenuItem : public sce::SampleUtil::Debug::AbstractMenuItem {};
```

Description

This is the debug menu item class that can manage (set/obtain/bind) unsigned 8-bit integer type values.

Methods Summary

Methods	Description
~Uint8MenuItem	Destructor.
getValue	Get unsigned 8-bit integer type value that is currently set.
setValue	Set unsigned 8-bit integer type value.
setReferenceValue	Bind unsigned 8-bit integer type variable to menu item.
downcast	Downcast function for Uint8MenuItem .

Constructors and Destructors

~Uint8MenuItem

Destructor.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Uint8MenuItem {
                virtual inline ~Uint8MenuItem();
            }
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Static Methods

downcast

Downcast function for [Uint8MenuItem](#).

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Uint8MenuItem {
                static Uint8MenuItem *downcast(
                    AbstractMenuItem *menuItem
                );
            };
        }
    }
}
```

Arguments

menuItem Pointer to [AbstractMenuItem](#)

Return Values

Value	Description
Uint8MenuItem*	If the downcast is successful, a pointer to Uint8MenuItem will be returned
NULL	If downcast failed

Description

This downcasts an [AbstractMenuItem](#) pointer to [Uint8MenuItem](#).

Public Instance Methods

getValue

Get unsigned 8-bit integer type value that is currently set.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class UInt8MenuItem {
                virtual uint8_t getValue(void)=0 const;
            }
        }
    }
}
```

Return Values

Current unsigned 8-bit integer type value

Description

This obtains the unsigned 8-bit integer type value that is currently set.

SCE CONFIDENTIAL

setValue

Set unsigned 8-bit integer type value.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class UInt8MenuItem {
                virtual void setValue(
                    uint8_t value
                )=0;
            }
        }
    }
}
```

Arguments

value Unsigned 8-bit integer type value

Return Values

None

Description

This sets the unsigned 8-bit integer type value.

SCE CONFIDENTIAL

setReferenceValue

Bind unsigned 8-bit integer type variable to menu item.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class UInt8MenuItem {
                virtual void setReferenceValue(
                    uint8_t *refValue
                )=0;
            }
        }
    }
}
```

Arguments

refValue Unsigned 8-bit integer type pointer

Return Values

None

Description

This binds an unsigned 8-bit integer type variable to a menu item.

sce::SampleUtil::Debug::Uint16Menulte

m

0000004892117

Summary

sce::SampleUtil::Debug::Uint16MenuItem

Debug menu item class that can manage unsigned 16-bit integer type values.

Definition

```
#include <menu.h>
class Uint16MenuItem : public sce::SampleUtil::Debug::AbstractMenuItem {};
```

Description

This is the debug menu item class that can manage (set/obtain/bind) unsigned 16-bit integer type values.

Methods Summary

Methods	Description
~Uint16MenuItem	Destructor.
getValue	Get unsigned 16-bit integer type value that is currently set.
setValue	Set unsigned 16-bit integer type value.
setReferenceValue	Bind unsigned 16-bit integer type variable to menu item.
downcast	Downcast function for Uint16MenuItem .

Constructors and Destructors

~Uint16MenuItem

Destructor.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Uint16MenuItem {
                virtual inline ~Uint16MenuItem();
            }
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Static Methods

downcast

Downcast function for [Uint16MenuItem](#).

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Uint16MenuItem {
                static Uint16MenuItem *downcast(
                    AbstractMenuItem *menuItem
                );
            };
        }
    }
}
```

Arguments

menuItem Pointer to [AbstractMenuItem](#)

Return Values

Value	Description
Uint16MenuItem*	If the downcast is successful, a pointer to Uint16MenuItem will be returned
NULL	If downcast failed

Description

This downcasts an [AbstractMenuItem](#) pointer to [Uint16MenuItem](#).

Public Instance Methods

getValue

Get unsigned 16-bit integer type value that is currently set.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class UInt16MenuItem {
                virtual uint16_t getValue(void)=0 const;
            }
        }
    }
}
```

Return Values

Current unsigned 16-bit integer type value

Description

This obtains the unsigned 16-bit integer type value that is currently set.

SCE CONFIDENTIAL

setValue

Set unsigned 16-bit integer type value.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Uint16MenuItem {
                virtual void setValue(
                    uint16_t value
                )=0;
            }
        }
    }
}
```

Arguments

value Unsigned 16-bit integer type value

Return Values

None

Description

This sets the unsigned 16-bit integer type value.

SCE CONFIDENTIAL

setReferenceValue

Bind unsigned 16-bit integer type variable to menu item.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class UInt16MenuItem {
                virtual void setReferenceValue(
                    uint16_t *refValue
                )=0;
            }
        }
    }
}
```

Arguments

refValue Unsigned 16-bit integer type pointer

Return Values

None

Description

This binds an unsigned 16-bit integer type variable to a menu item.

sce::SampleUtil::Debug::Uint32Menulte

m

0000004892117

Summary

sce::SampleUtil::Debug::Uint32MenuItem

Debug menu item class that can manage unsigned 32-bit integer type values.

Definition

```
#include <menu.h>
class Uint32MenuItem : public sce::SampleUtil::Debug::AbstractMenuItem {};
```

Description

This is the debug menu item class that can manage (set/obtain/bind) unsigned 32-bit integer type values.

Methods Summary

Methods	Description
~Uint32MenuItem	Destructor.
getValue	Get unsigned 32-bit integer type value that is currently set.
setValue	Set unsigned 32-bit integer type value.
setReferenceValue	Bind unsigned 32-bit integer type variable to menu item.
downcast	Downcast function for Uint32MenuItem .

Constructors and Destructors

~Uint32MenuItem

Destructor.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Uint32MenuItem {
                virtual inline ~Uint32MenuItem();
            }
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Static Methods

downcast

Downcast function for [Uint32MenuItem](#).

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Uint32MenuItem {
                static Uint32MenuItem *downcast(
                    AbstractMenuItem *menuItem
                );
            };
        }
    }
}
```

Arguments

menuItem Pointer to [AbstractMenuItem](#)

Return Values

Value	Description
Uint32MenuItem*	If the downcast is successful, a pointer to Uint32MenuItem will be returned
NULL	If downcast failed

Description

This downcasts an [AbstractMenuItem](#) pointer to [Uint32MenuItem](#).

Public Instance Methods

getValue

Get unsigned 32-bit integer type value that is currently set.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class UInt32MenuItem {
                virtual uint32_t getValue(void)=0 const;
            }
        }
    }
}
```

Return Values

Current unsigned 32-bit integer type value

Description

This obtains the unsigned 32-bit integer type value that is currently set.

SCE CONFIDENTIAL

setValue

Set unsigned 32-bit integer type value.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class UInt32MenuItem {
                virtual void setValue(
                    uint32_t value
                )=0;
            }
        }
    }
}
```

Arguments

value Unsigned 32-bit integer type value

Return Values

None

Description

This sets the unsigned 32-bit integer type value.

SCE CONFIDENTIAL

setReferenceValue

Bind unsigned 32-bit integer type variable to menu item.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class UInt32MenuItem {
                virtual void setReferenceValue(
                    uint32_t *refValue
                )=0;
            }
        }
    }
}
```

Arguments

refValue Unsigned 32-bit integer type pointer

Return Values

None

Description

This binds an unsigned 32-bit integer type variable to a menu item.

sce::SampleUtil::Debug::Uint64Menulte

m

0000004892117

Summary

sce::SampleUtil::Debug::Uint64MenuItem

Debug menu item class that can manage unsigned 64-bit integer type values.

Definition

```
#include <menu.h>
class Uint64MenuItem : public sce::SampleUtil::Debug::AbstractMenuItem {};
```

Description

This is the debug menu item class that can manage (set/obtain/bind) unsigned 64-bit integer type values.

Methods Summary

Methods	Description
~Uint64MenuItem	Destructor.
getValue	Get unsigned 64-bit integer type value that is currently set.
setValue	Set unsigned 64-bit integer type value.
setReferenceValue	Bind unsigned 64-bit integer type variable to menu item.
downcast	Downcast function for Uint64MenuItem .

Constructors and Destructors

~Uint64MenuItem

Destructor.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Uint64MenuItem {
                virtual inline ~Uint64MenuItem();
            }
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Static Methods

downcast

Downcast function for [Uint64MenuItem](#).

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Uint64MenuItem {
                static Uint64MenuItem *downcast(
                    AbstractMenuItem *menuItem
                );
            };
        }
    }
}
```

Arguments

menuItem Pointer to [AbstractMenuItem](#)

Return Values

Value	Description
Uint64MenuItem*	If the downcast is successful, a pointer to Uint64MenuItem will be returned
NULL	If downcast failed

Description

This downcasts an [AbstractMenuItem](#) pointer to [Uint64MenuItem](#).

Public Instance Methods

getValue

Get unsigned 64-bit integer type value that is currently set.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class UInt64MenuItem {
                virtual uint64_t getValue(void)=0 const;
            }
        }
    }
}
```

Return Values

Current unsigned 64-bit integer type value

Description

This obtains the unsigned 64-bit integer type value that is currently set.

SCE CONFIDENTIAL

setValue

Set unsigned 64-bit integer type value.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class UInt64MenuItem {
                virtual void setValue(
                    uint64_t value
                )=0;
            }
        }
    }
}
```

Arguments

value Unsigned 64-bit integer type value

Return Values

None

Description

This sets the unsigned 64-bit integer type value.

SCE CONFIDENTIAL

setReferenceValue

Bind unsigned 64-bit integer type variable to menu item.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class UInt64MenuItem {
                virtual void setReferenceValue(
                    uint64_t *refValue
                )=0;
            }
        }
    }
}
```

Arguments

refValue Unsigned 64-bit integer type pointer

Return Values

None

Description

This binds an unsigned 64-bit integer type variable to a menu item.

sce::SampleUtil::Debug::FloatMenuItem

Summary

sce::SampleUtil::Debug::FloatMenuItem

Debug menu item class that can manage single precision floating-point type values.

Definition

```
#include <menu.h>
class FloatMenuItem : public sce::SampleUtil::Debug::AbstractMenuItem {};
```

Description

This is the debug menu item class that can manage (set/obtain/bind) single precision floating-point type values.

Methods Summary

Methods	Description
<u>~FloatMenuItem</u>	Destructor.
<u>getValue</u>	Get current single precision floating-point type value that is set.
<u>setValue</u>	Set single precision floating-point type value.
<u>setReferenceValue</u>	Bind single precision floating-point type value variable to menu item.
<u>downcast</u>	Downcast function for <u>FloatMenuItem</u> .

Constructors and Destructors

~FloatMenuItem

Destructor.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class FloatMenuItem {
                virtual inline ~FloatMenuItem();
            }
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Static Methods

downcast

Downcast function for [FloatMenuItem](#).

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class FloatMenuItem {
                static FloatMenuItem *downcast(
                    AbstractMenuItem *menuItem
                );
            };
        }
    }
}
```

Arguments

menuItem Pointer to [AbstractMenuItem](#)

Return Values

Value	Description
FloatMenuItem*	If the downcast is successful, a pointer to FloatMenuItem will be returned
NULL	If downcast failed

Description

Downcasts an [AbstractMenuItem](#) pointer to [FloatMenuItem](#).

Public Instance Methods

getValue

Get current single precision floating-point type value that is set.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class FloatMenuItem {
                virtual float getValue(void)=0 const;
            }
        }
    }
}
```

Return Values

Current single precision floating-point value

Description

This obtains the current single precision floating-point type value that is set.

SCE CONFIDENTIAL

setValue

Set single precision floating-point type value.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class FloatMenuItem {
                virtual void setValue(
                    float value
                )=0;
            }
        }
    }
}
```

Arguments

value Single precision floating-point type value

Return Values

None

Description

This sets the single precision floating-point type value.

SCE CONFIDENTIAL

setReferenceValue

Bind single precision floating-point type value variable to menu item.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class FloatMenuItem {
                virtual void setReferenceValue(
                    float *refValue
                )=0;
            }
        }
    }
}
```

Arguments

refValue Single precision floating-point type value pointer

Return Values

None

Description

This binds a single precision floating-point type value variable to a menu item.

sce::SampleUtil::Debug::DoubleMenulte

m

000004892117

Summary

sce::SampleUtil::Debug::DoubleMenuItem

Debug menu item class that can manage double precision floating-point type values.

Definition

```
#include <menu.h>
class DoubleMenuItem : public sce::SampleUtil::Debug::AbstractMenuItem {};
```

Description

This is the debug menu item class that can manage (set/obtain/bind) double precision floating-point type values.

Methods Summary

Methods	Description
~DoubleMenuItem	Destructor.
getValue	Get current double precision floating-point type value that is set.
setValue	Set double precision floating-point type value.
setReferenceValue	Bind double precision floating-point type value variable to menu item.
downcast	Cast to DoubleMenuItem .

Constructors and Destructors

~DoubleMenuItem

Destructor.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class DoubleMenuItem {
                virtual inline ~DoubleMenuItem();
            }
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Static Methods

downcast

Cast to [DoubleMenuItem](#).

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class DoubleMenuItem {
                static DoubleMenuItem *downcast(
                    AbstractMenuItem *menuItem
                );
            };
        }
    }
}
```

Arguments

menuItem Pointer to [AbstractMenuItem](#)

Return Values

Value	Description
DoubleMenuItem*	If the downcast is successful, a pointer to Int64MenuItem will be returned
NULL	If downcast failed

Description

This casts to [DoubleMenuItem](#).

Public Instance Methods

getValue

Get current double precision floating-point type value that is set.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class DoubleMenuItem {
                virtual double getValue(void)=0 const;
            }
        }
    }
}
```

Return Values

Current double precision floating-point value

Description

This obtains the current double precision floating-point type value that is set.

SCE CONFIDENTIAL

setValue

Set double precision floating-point type value.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class DoubleMenuItem {
                virtual void setValue(
                    double value
                )=0;
            }
        }
    }
}
```

Arguments

value Double precision floating-point type value

Return Values

None

Description

This sets the double precision floating-point type value.

SCE CONFIDENTIAL

setReferenceValue

Bind double precision floating-point type value variable to menu item.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class DoubleMenuItem {
                virtual void setReferenceValue(
                    double *refValue
                )=0;
            }
        }
    }
}
```

Arguments

refValue Double precision floating-point type value pointer

Return Values

None

Description

This binds a double precision floating-point type value variable to a menu item.

sce::SampleUtil::Debug::BoolMenuItem

Summary

sce::SampleUtil::Debug::BoolMenuItem

Debug menu item class that can manage Bool type values.

Definition

```
#include <menu.h>
class BoolMenuItem : public sce::SampleUtil::Debug::AbstractMenuItem {};
```

Description

This debug menu item class that can manage (set/obtain/bind) Bool type values.

Methods Summary

Methods	Description
~BoolMenuItem	Destructor.
getValue	Obtains the current Bool type value that is set.
setValue	Sets Bool type value.
setReferenceValue	Binds Bool type variable to menu item.
downcast	Downcast function for BoolMenuItem .

Constructors and Destructors

~BoolMenuItem

Destructor.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class BoolMenuItem {
                virtual inline ~BoolMenuItem();
            }
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Static Methods

downcast

Downcast function for [BoolMenuItem](#).

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class BoolMenuItem {
                static BoolMenuItem *downcast(
                    AbstractMenuItem *menuItem
                );
            };
        }
    }
}
```

Arguments

menuItem Pointer to [AbstractMenuItem](#)

Return Values

Value	Description
BoolMenuItem*	If the downcast is successful, a pointer to BoolMenuItem will be returned
NULL	If downcast failed

Description

Downcasts an [AbstractMenuItem](#) pointer to [BoolMenuItem](#).

Public Instance Methods

getValue

Obtains the current Bool type value that is set.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class BoolMenuItem {
                virtual bool getValue(void)=0 const;
            }
        }
    }
}
```

Return Values

Current Bool type value

Description

This obtains the current Bool type value that is set.

SCE CONFIDENTIAL

setValue

Sets Bool type value.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class BoolMenuItem {
                virtual void setValue(
                    bool value
                )=0;
            }
        }
    }
}
```

Arguments

value Bool type value

Return Values

None

Description

Sets the Bool type value.

SCE CONFIDENTIAL

setReferenceValue

Binds Bool type variable to menu item.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class BoolMenuItem {
                virtual void setReferenceValue(
                    bool *refValue
                )=0;
            }
        }
    }
}
```

Arguments

refValue Bool type pointer

Return Values

None

Description

This binds a Bool type variable to a menu item.

sce::SampleUtil::Debug::EnumMenulte

m

000004892117

Summary

sce::SampleUtil::Debug::EnumMenuItem

Debug menu item class that can manage indexed character string set types.

Definition

```
#include <menu.h>
class EnumMenuItem : public sce::SampleUtil::Debug::AbstractMenuItem {};
```

Description

This is the debug menu item class that can manage (set/obtain/bind) indexed character string set types

Methods Summary

Methods	Description
<u>~EnumMenuItem</u>	Destructor.
<u>getValue</u>	Obtains the current Enum index value that is set.
<u>setValue</u>	Sets Enum index value.
<u>setReferenceValue</u>	Binds int type variable to Enum index value held by menu item.
<u>downcast</u>	Downcast function for <u>EnumMenuItem</u> .

Constructors and Destructors

~EnumMenuItem

Destructor.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class EnumMenuItem {
                virtual inline ~EnumMenuItem();
            };
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Static Methods

downcast

Downcast function for [EnumMenuItem](#).

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class EnumMenuItem {
                static EnumMenuItem *downcast(
                    AbstractMenuItem *menuItem
                );
            };
        }
    }
}
```

Arguments

menuItem Pointer to [AbstractMenuItem](#)

Return Values

Value	Description
EnumMenuItem*	If the downcast is successful, a pointer to EnumMenuItem will be returned
NULL	If downcast failed

Description

Downcasts an [AbstractMenuItem](#) pointer to [EnumMenuItem](#).

Public Instance Methods

getValue

Obtains the current Enum index value that is set.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class EnumMenuItem {
                virtual int getValue(void)=0 const;
            }
        }
    }
}
```

Return Values

Current Enum index value

Description

This obtains the current Enum index value that is set.

SCE CONFIDENTIAL

setValue

Sets Enum index value.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class EnumMenuItem {
                virtual void setValue(
                    int value
                )=0;
            }
        }
    }
}
```

Arguments

value Enum index value

Return Values

None

Description

Sets the Enum index value.

SCE CONFIDENTIAL

setReferenceValue

Binds int type variable to Enum index value held by menu item.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class EnumMenuItem {
                virtual void setReferenceValue(
                    int *refValue
                )=0;
            }
        }
    }
}
```

Arguments

refValue int type pointer

Return Values

None

Description

Binds an int type variable to an Enum index value held by a menu item.

sce::SampleUtil::Debug::MenuItemGrou

p

Summary

sce::SampleUtil::Debug::MenuItemGroup

Debug menu item class that can group multiple AbstractMenuItems.

Definition

```
#include <menu.h>
class MenuItemGroup : public sce::SampleUtil::Debug::AbstractMenuItem {};
```

Description

This debug menu item class can group multiple AbstractMenuItems.

Methods Summary

Methods	Description
~MenuItemGroup	Destructor.
getItemById	Obtains pointer from id to AbstractMenuItem .
getItemByIndex	Obtains pointer from index to AbstractMenuItem .
getNumItems	Obtains the number of menu items belonging to this group.
deleteItemById	Deletes menu item from id.
deleteItemByIndex	Deletes menu item from index.
addInt8MenuItem	Add menu item including signed 8-bit integer type value.
addInt16MenuItem	Add menu item including signed 16-bit integer type value.
addInt32MenuItem	Add menu item including signed 32-bit integer type value.
addIntMenuItem	Add menu item including signed 32-bit integer type value.
addInt64MenuItem	Add menu item including signed 64-bit integer type value.
addUInt8MenuItem	Add menu item including unsigned 8-bit integer type value.
addUInt16MenuItem	Add menu item including unsigned 16-bit integer type value.
addUInt32MenuItem	Add menu item including unsigned 32-bit integer type value.
addUInt64MenuItem	Add menu item including unsigned 64-bit integer type value.
addFloatMenuItem	Add menu item including single precision floating-point type value.
addDoubleMenuItem	Add menu item including double precision floating-point type value.
addBoolMenuItem	Add menu item that includes a bool type value.
addEnumMenuItem	Add menu item that includes indexed character string set type value.
addMenuItemGroup	Adds menu item group.
setLabel	Sets label.
downcast	Downcast function for MenuItemGroup .

Constructors and Destructors

~MenuItemGroup

Destructor.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class MenuItemGroup {
                virtual inline ~MenuItemGroup();
            }
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Static Methods

downcast

Downcast function for [MenuItemGroup](#).

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class MenuItemGroup {
                static MenuItemGroup *downcast(
                    AbstractMenuItem *menuItem
                );
            };
        }
    }
}
```

Arguments

menuItem Pointer to [AbstractMenuItem](#)

Return Values

Value	Description
MenuItemGroup*	If the downcast is successful, a pointer to MenuItemGroup will be returned
NULL	If downcast failed

Description

Downcasts an [AbstractMenuItem](#) pointer to [MenuItemGroup](#).

Public Instance Methods

getItemById

Obtains pointer from id to [AbstractMenuItem](#).

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class MenuItemGroup {
                virtual AbstractMenuItem *getItemById(
                    const char *id
                )=0 const;
            }
        }
    }
}
```

Arguments

id [Menu](#) item id

Return Values

Value	Description
AbstractMenuItem*	When AbstractMenuItem linked to id is found, a pointer to AbstractMenuItem will be returned
NULL	When AbstractMenuItem linked to id could not be found

Description

Searches for a pointer to [AbstractMenuItem](#) linked to id and obtains it.

getItemByIndex

Obtains pointer from index to [AbstractMenuItem](#).

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class MenuItemGroup {
                virtual AbstractMenuItem * getItemByIndex(
                    uint32_t index
                )=0 const;
            }
        }
    }
}
```

Arguments

index [Menu](#) item index

Return Values

Value	Description
AbstractMenuItem*	When index # AbstractMenuItem is found, a pointer to AbstractMenuItem will be returned
NULL	When index # AbstractMenuItem could not be found

Description

Searches for a pointer to index # [AbstractMenuItem](#) and obtains it.

SCE CONFIDENTIAL

getNumItems

Obtains the number of menu items belonging to this group.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class MenuItemGroup {
                virtual uint32_t getNumItems(void)=0 const;
            }
        }
    }
}
```

Return Values

Number of menu items

Description

This obtains the number of menu items belonging to this group.

SCE CONFIDENTIAL

deleteItemById

Deletes menu item from id.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class MenuItemGroup {
                virtual void deleteItemId(
                    const char *id
                )=0;
            }
        }
    }
}
```

Arguments

id [Menu](#) item id

Return Values

None

Description

Searches for a menu item linked to id and deletes it if found

SCE CONFIDENTIAL

deleteItemByIndex

Deletes menu item from index.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class MenuItemGroup {
                virtual int deleteItemByIndex(
                    uint32_t index
                )=0;
            }
        }
    }
}
```

Arguments

index [Menu](#) item index

Return Values

Value	Description
SCE_OK	Deletion successful
SCE_SAMPLE_UTIL_ERROR_OUT_OF_MEMORY	When index # menu item could not be found

Description

Searches for a menu item linked to the index # and deletes it if found

SCE CONFIDENTIAL

addInt8MenuItem

Add menu item including signed 8-bit integer type value.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class MenuItemGroup {
                virtual Int8MenuItem *addInt8MenuItem(
                    const char *id,
                    const char *label,
                    int8_t value = 0,
                    int8_t step = 1,
                    int8_t minValue = SCHAR_MIN,
                    int8_t maxValue = SCHAR_MAX,
                    int position = -1,
                    const char *format = NULL
                )=0;
            }
        }
    }
}
```

Arguments

<i>id</i>	Menu item id
<i>label</i>	Menu item descriptor
<i>value</i>	Menu item value (default: 0)
<i>step</i>	Value step (default: 1)
<i>minValue</i>	Minimum value available (default: SCHAR_MIN)
<i>maxValue</i>	Maximum value available (default: SCHAR_MAX)
<i>position</i>	Menu item position (default: -1)
<i>format</i>	User-defined formatting

Return Values

Pointer to added [Int8MenuItem](#) instance

Description

This adds a menu item including a signed 8-bit integer type value to a menu. By setting printf format formatting to the format argument, a value can be stringized with an arbitrary format. A specifier (%d) must be inserted in the formatting in order to stringize the value.

SCE CONFIDENTIAL

addInt16MenuItem

Add menu item including signed 16-bit integer type value.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class MenuItemGroup {
                virtual Int16MenuItem *addInt16MenuItem(
                    const char *id,
                    const char *label,
                    int16_t value = 0,
                    int16_t step = 1,
                    int16_t minValue = SHRT_MIN,
                    int16_t maxValue = SHRT_MAX,
                    int position = -1,
                    const char *format = NULL
                )=0;
            }
        }
    }
}
```

Arguments

<i>id</i>	Menu item id
<i>label</i>	Menu item descriptor
<i>value</i>	Menu item value (default: 0)
<i>step</i>	Value step (default: 1)
<i>minValue</i>	Minimum value available (default: SHRT_MIN)
<i>maxValue</i>	Maximum value available (default: SHRT_MAX)
<i>position</i>	Menu item position (default: -1)
<i>format</i>	User-defined formatting

Return Values

Pointer to added [Int16MenuItem](#) instance

Description

This adds a menu item including a signed 16-bit integer type value to a menu. By setting printf format formatting to the format argument, a value can be stringized with an arbitrary format. A specifier (%d) must be inserted in the formatting in order to stringize the value.

SCE CONFIDENTIAL

addInt32MenuItem

Add menu item including signed 32-bit integer type value.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class MenuItemGroup {
                virtual Int32MenuItem *addInt32MenuItem(
                    const char *id,
                    const char *label,
                    int32_t value = 0,
                    int32_t step = 1,
                    int32_t minValue = INT_MIN,
                    int32_t maxValue = INT_MAX,
                    int position = -1,
                    const char *format = NULL
                )=0;
            }
        }
    }
}
```

Arguments

<i>id</i>	Menu item id
<i>label</i>	Menu item descriptor
<i>value</i>	Menu item value (default: 0)
<i>step</i>	Value step (default: 1)
<i>minValue</i>	Minimum value available (default: INT_MIN)
<i>maxValue</i>	Maximum value available (default: INT_MAX)
<i>position</i>	Menu item position (default: -1)
<i>format</i>	User-defined formatting

Return Values

Pointer to added [Int32MenuItem](#) instance

Description

This adds a menu item including a signed 32-bit integer type value to a menu. By setting printf format formatting to the format argument, a value can be stringized with an arbitrary format. A specifier (%d) must be inserted in the formatting in order to stringize the value.

SCE CONFIDENTIAL

addIntMenuItem

Add menu item including signed 32-bit integer type value.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class MenuItemGroup {
                inline IntMenuItem *addIntMenuItem(
                    const char *id,
                    const char *label,
                    int32_t value = 0,
                    int32_t step = 1,
                    int32_t minValue = INT_MIN,
                    int32_t maxValue = INT_MAX,
                    int position = -1,
                    const char *format = NULL
                );
            };
        }
    }
}
```

Arguments

<i>id</i>	Menu item id
<i>label</i>	Menu item descriptor
<i>value</i>	Menu item value (default: 0)
<i>step</i>	Value step (default: 1)
<i>minValue</i>	Minimum value available (default: INT_MIN)
<i>maxValue</i>	Maximum value available (default: INT_MAX)
<i>position</i>	Menu item position (default: -1)
<i>format</i>	User-defined formatting

Return Values

Pointer to added IntMenuItem instance

Description

This adds a menu item including a signed 32-bit integer type value to a menu. By setting printf format formatting to the format argument, a value can be stringized with an arbitrary format. A specifier (%d) must be inserted in the formatting in order to stringize the value.

addInt64MenuItem

Add menu item including signed 64-bit integer type value.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class MenuItemGroup {
                virtual Int64MenuItem *addInt64MenuItem(
                    const char *id,
                    const char *label,
                    int64_t value = 0,
                    int64_t step = 1,
                    int64_t minValue = LLONG_MIN,
                    int64_t maxValue = LLONG_MAX,
                    int position = -1,
                    const char *format = NULL
                )=0;
            }
        }
    }
}
```

Arguments

<i>id</i>	Menu item id
<i>label</i>	Menu item descriptor
<i>value</i>	Menu item value (default: 0)
<i>step</i>	Value step (default: 1)
<i>minValue</i>	Minimum value available (default: LLONG_MIN)
<i>maxValue</i>	Maximum value available (default: LLONG_MAX)
<i>position</i>	Menu item position (default: -1)
<i>format</i>	User-defined formatting

Return Values

Pointer to added [Int64MenuItem](#) instance

Description

This adds a menu item including a signed 64-bit integer type value to a menu. By setting printf format formatting to the format argument, a value can be stringized with an arbitrary format. A specifier (%ld) must be inserted in the formatting in order to stringize the value.

SCE CONFIDENTIAL

addUInt8MenuItem

Add menu item including unsigned 8-bit integer type value.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class MenuItemGroup {
                virtual UInt8MenuItem *addUInt8MenuItem(
                    const char *id,
                    const char *label,
                    uint8_t value = 0,
                    uint8_t step = 1,
                    uint8_t minValue = 0,
                    uint8_t maxValue = UCHAR_MAX,
                    int position = -1,
                    const char *format = NULL
                )=0;
            }
        }
    }
}
```

Arguments

<i>id</i>	Menu item id
<i>label</i>	Menu item descriptor
<i>value</i>	Menu item value (default: 0)
<i>step</i>	Value step (default: 1)
<i>minValue</i>	Minimum value available (default: 0)
<i>maxValue</i>	Maximum value available (default: UCHAR_MAX)
<i>position</i>	Menu item position (default: -1)
<i>format</i>	User-defined formatting

Return Values

Pointer to added [UInt8MenuItem](#) instance

Description

This adds a menu item including an unsigned 8-bit integer type value to a menu. By setting printf format formatting to the format argument, a value can be stringized with an arbitrary format. A specifier (%d or %x) must be inserted in the formatting in order to stringize the value.

SCE CONFIDENTIAL

addUint16MenuItem

Add menu item including unsigned 16-bit integer type value.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class MenuItemGroup {
                virtual Uint16MenuItem *addUint16MenuItem(
                    const char *id,
                    const char *label,
                    uint16_t value = 0,
                    uint16_t step = 1,
                    uint16_t minValue = 0,
                    uint16_t maxValue = USHRT_MAX,
                    int position = -1,
                    const char *format = NULL
                )=0;
            }
        }
    }
}
```

Arguments

<i>id</i>	Menu item id
<i>label</i>	Menu item descriptor
<i>value</i>	Menu item value (default: 0)
<i>step</i>	Value step (default: 1)
<i>minValue</i>	Minimum value available (default: 0)
<i>maxValue</i>	Maximum value available (default: USHRT_MAX)
<i>position</i>	Menu item position (default: -1)
<i>format</i>	User-defined formatting

Return Values

Pointer to added [Uint16MenuItem](#) instance

Description

This adds a menu item including an unsigned 16-bit integer type value to a menu. By setting printf format formatting to the format argument, a value can be stringized with an arbitrary format. A specifier (%d or %x) must be inserted in the formatting in order to stringize the value.

SCE CONFIDENTIAL

addUInt32MenuItem

Add menu item including unsigned 32-bit integer type value.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class MenuItemGroup {
                virtual UInt32MenuItem *addUInt32MenuItem(
                    const char *id,
                    const char *label,
                    uint32_t value = 0,
                    uint32_t step = 1,
                    uint32_t minValue = 0,
                    uint32_t maxValue = UINT_MAX,
                    int position = -1,
                    const char *format = NULL
                )=0;
            }
        }
    }
}
```

Arguments

<i>id</i>	Menu item id
<i>label</i>	Menu item descriptor
<i>value</i>	Menu item value (default: 0)
<i>step</i>	Value step (default: 1)
<i>minValue</i>	Minimum value available (default: 0)
<i>maxValue</i>	Maximum value available (default: <code>UINT_MAX</code>)
<i>position</i>	Menu item position (default: -1)
<i>format</i>	User-defined formatting

Return Values

Pointer to added [UInt32MenuItem](#) instance

Description

This adds a menu item including an unsigned 32-bit integer type value to a menu. By setting printf format formatting to the format argument, a value can be stringized with an arbitrary format. A specifier (%d or %x) must be inserted in the formatting in order to stringize the value.

SCE CONFIDENTIAL

addUint64MenuItem

Add menu item including unsigned 64-bit integer type value.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class MenuItemGroup {
                virtual Uint64MenuItem *addUint64MenuItem(
                    const char *id,
                    const char *label,
                    uint64_t value = 0,
                    uint64_t step = 1,
                    uint64_t minValue = 0,
                    uint64_t maxValue = ULONG_MAX,
                    int position = -1,
                    const char *format = NULL
                )=0;
            }
        }
    }
}
```

Arguments

<i>id</i>	Menu item id
<i>label</i>	Menu item descriptor
<i>value</i>	Menu item value (default: 0)
<i>step</i>	Value step (default: 1)
<i>minValue</i>	Minimum value available (default: 0)
<i>maxValue</i>	Maximum value available (default: ULONG_MAX)
<i>position</i>	Menu item position (default: -1)
<i>format</i>	User-defined formatting

Return Values

Pointer to added [Uint64MenuItem](#) instance

Description

This adds a menu item including an unsigned 64-bit integer type value to a menu. By setting printf format formatting to the format argument, a value can be stringized with an arbitrary format. A specifier (%ld or %lx) must be inserted in the formatting in order to stringize the value.

addFloatMenuItem

Add menu item including single precision floating-point type value.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class MenuItemGroup {
                virtual FloatMenuItem *addFloatMenuItem(
                    const char *id,
                    const char *label,
                    float value = 0.0f,
                    float step = 0.1f,
                    float minValue = -FLT_MAX,
                    float maxValue = FLT_MAX,
                    int position = -1,
                    const char *format = NULL
                )=0;
            }
        }
    }
}
```

Arguments

<i>id</i>	Menu item id
<i>label</i>	Menu item descriptor
<i>value</i>	Menu item value (default: 0.0f)
<i>step</i>	Value step (default: 0.1f)
<i>minValue</i>	Minimum value available (default: -FLT_MAX)
<i>maxValue</i>	Maximum value available (default: FLT_MAX)
<i>position</i>	Menu item position (default: -1)
<i>format</i>	User-defined formatting

Return Values

Pointer to added [FloatMenuItem](#) instance

Description

This adds a menu item including a single precision floating-point type value to a menu. By setting printf format formatting to the format argument, a value can be stringized with an arbitrary format. A specifier (%f) must be inserted in the formatting in order to stringize the value.

SCE CONFIDENTIAL

addDoubleMenuItem

Add menu item including double precision floating-point type value.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class MenuItemGroup {
                virtual DoubleMenuItem *addDoubleMenuItem(
                    const char *id,
                    const char *label,
                    double value = 0.0f,
                    double step = 0.1f,
                    double minValue = -DBL_MAX,
                    double maxValue = DBL_MAX,
                    int position = -1,
                    const char *format = NULL
                )=0;
            }
        }
    }
}
```

Arguments

<i>id</i>	Menu item id
<i>label</i>	Menu item descriptor
<i>value</i>	Menu item value (default: 0.0f)
<i>step</i>	Value step (default: 0.1f)
<i>minValue</i>	Minimum value available (default: -DBL_MAX)
<i>maxValue</i>	Maximum value available (default: DBL_MAX)
<i>position</i>	Menu item position (default: -1)
<i>format</i>	User-defined formatting

Return Values

Pointer to added [DoubleMenuItem](#) instance

Description

This adds a menu item including a single precision floating-point type value to a menu. By setting printf format formatting to the format argument, a value can be stringized with an arbitrary format. A specifier (%lf) must be inserted in the formatting in order to stringize the value.

SCE CONFIDENTIAL

addBoolMenuItem

Add menu item that includes a bool type value.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class MenuItemGroup {
                virtual BoolMenuItem *addBoolMenuItem(
                    const char *id,
                    const char *label,
                    bool value = true,
                    int position = -1,
                    const char *format = NULL
                )=0;
            }
        }
    }
}
```

Arguments

<i>id</i>	Menu item id
<i>label</i>	Menu item descriptor
<i>value</i>	Menu item value (default: true)
<i>position</i>	Menu item position (default: -1)
<i>format</i>	User-defined formatting

Return Values

Pointer to added [BoolMenuItem](#) instance

Description

This adds a menu item that includes a bool type value to a menu. By setting printf format formatting to the format argument, a value can be stringized with an arbitrary format. A specifier (%s) must be inserted in the formatting in order to stringize the value (true/false are displayed as their respective character strings).

addEnumMenuItem

Add menu item that includes indexed character string set type value.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class MenuItemGroup {
                virtual EnumMenuItem *addEnumMenuItem(
                    const char *id,
                    const char *label,
                    const char *enumLabels[],
                    uint32_t numLabels,
                    int value = 0,
                    int position = -1,
                    const char *format = NULL
                )=0;
            }
        }
    }
}
```

Arguments

<i>id</i>	Menu item id
<i>label</i>	Menu item descriptor
<i>enumLabels</i>	Character string to be used for the menu item value
<i>numLabels</i>	Number of character strings included in enumLabel
<i>value</i>	Menu item value (default: 0)
<i>position</i>	Menu item position (default: -1)
<i>format</i>	User-defined formatting

Return Values

Pointer to added [EnumMenuItem](#) instance

Description

This adds a menu item that includes an indexed character string set type value to a menu. By setting printf format formatting to the format argument, a value can be stringized with an arbitrary format. A specifier (%s) must be inserted in the formatting in order to stringize the value (character strings for indices will be displayed).

SCE CONFIDENTIAL

addMenuItemGroup

Adds menu item group.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class MenuItemGroup {
                virtual MenuItemGroup *addMenuItemGroup(
                    const char *id,
                    const char *label,
                    int position = -1
                )=0;
            }
        }
    }
}
```

Arguments

<i>id</i>	MenuItem item id
<i>label</i>	MenuItem item descriptor
<i>position</i>	MenuItem item position (default: -1)

Return Values

Pointer to added [MenuItemGroup](#) instance

Description

Adds a menu item group.

SCE CONFIDENTIAL

setLabel

Sets label.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class MenuItemGroup {
                virtual void setLabel(
                    const char *label
                )=0;
        }
    }
}
```

Arguments

label [Menu](#) item descriptor

Return Values

None

Description

This sets a label.

sce::SampleUtil::Debug::Menu

Summary

sce::SampleUtil::Debug::Menu

Class that performs management/rendering/etc. for a menu that is a tree structure.

Definition

```
#include <menu.h>
class Menu : public sce::SampleUtil::Resource {};
```

Description

This class performs management/rendering/etc. for a menu that is a tree structure.

Methods Summary

Methods	Description
~Menu	Destructor.
getRoot	Obtains the root of the managed menu tree.
findItem	Obtains pointer from id to AbstractMenuItem .
draw	Draw menu item.
update	Menu update.
update	Menu update.

Constructors and Destructors

~Menu

Destructor.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Menu {
                virtual inline ~Menu();
            }
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Public Instance Methods

getRoot

Obtains the root of the managed menu tree.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Menu {
                virtual MenuItemGroup *getRoot(void)=0 const;
            }
        }
    }
}
```

Return Values

Pointer to [MenuItemGroup](#)

Description

This obtains the root of the managed menu tree.

SCE CONFIDENTIAL

findItem

Obtains pointer from id to [AbstractMenuItem](#).

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Menu {
                virtual AbstractMenuItem *findItem(
                    const char *id
                )=0 const;
            }
        }
    }
}
```

Arguments

id [Menu](#) item id that also includes a group id

Return Values

Value	Description
AbstractMenuItem*	When a menu item linked to id is found, a pointer to AbstractMenuItem will be returned
NULL	When menu item linked to id could not be found

Description

Obtains a pointer to the [AbstractMenuItem](#) linked to id. This id includes a group id unlike getitemby.

SCE CONFIDENTIAL

draw

Draw menu item.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Menu {
                virtual int draw(
                    sce::SampleUtil::Graphics::GraphicsContext *gfxContext,
                    sce::SampleUtil::Graphics::SpriteRenderer *spriteRenderer,
                    uint32_t xInPix,
                    uint32_t yInPix
                )=0;
            };
        }
    }
}
```

Arguments

<i>gfxContext</i>	Pointer to GraphicsContext
<i>spriteRenderer</i>	Pointer to SpriteRenderer
<i>xInPix</i>	x coordinate in pixels
<i>yInPix</i>	y coordinate in pixels

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Failure (error code)

Description

Renders a menu item.

SCE CONFIDENTIAL

update

Menu update.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Menu {
                virtual void update(
                    const Input::ControllerContext *ctrl
                )=0;
            }
        }
    }
}
```

Arguments

ctrl Pointer to PadContext

Return Values

None

Description

Updates the status of all of the menu items included in Menu

SCE CONFIDENTIAL

update

Menu update.

Definition

```
#include <menu.h>
namespace sce {
    namespace SampleUtil {
        namespace Debug {
            class Menu {
                virtual void update(
                    const Input::PadContext *pad
                )=0;
            }
        }
    }
}
```

Arguments

pad Pointer to PadContext

Return Values

None

Description

Updates the status of all of the menu items included in Menu

SCE CONFIDENTIAL

sce::SampleUtil::Graphics

000004892117

Summary

sce::SampleUtil::Graphics

Graphics-associated definitions.

Definition

```
namespace Graphics { }
```

Description

These are the Graphics-associated definitions.

Function Summary

Function	Description
createGraphicsContext	Generate graphics context.
createFontLoader	Create FontLoader class.
createSpriteRenderer	Create SpriteRenderer .
createObject3dRenderer	Generate Object3dRenderer .

Inner Classes, Structures, and Namespaces

Item	Description
sce::SampleUtil::Graphics::BufferInterface	Class which becomes the base of a buffer object.
sce::SampleUtil::Graphics::Buffer	One-dimensional buffer class.
sce::SampleUtil::Graphics::IndexBuffer	Index buffer.
sce::SampleUtil::Graphics::VertexBuffer	Vertex buffer.
sce::SampleUtil::Graphics::UniformBuffer	Uniform buffer.
sce::SampleUtil::Graphics::StructuredBuffer	Structured buffer.
sce::SampleUtil::Graphics::Texture	Texture buffer class.
sce::SampleUtil::Graphics::RenderTarget	Render target.
sce::SampleUtil::Graphics::DepthStencilSurface	Depth stencil target.
sce::SampleUtil::Graphics::BlendInfo	The description about blending and masking to the color surface of a shader result.
sce::SampleUtil::Graphics::GraphicsContextOption	Structure for initializing GraphicsContext .
sce::SampleUtil::Graphics::GraphicsContext	Graphics context.
sce::SampleUtil::Graphics::Effect	Name space associated with the effect.
sce::SampleUtil::Graphics::FontParam	Structure used to initialize the Font class.
sce::SampleUtil::Graphics::Font	Class to handle the font texture.
sce::SampleUtil::Graphics::FontLoader	Class that generates fonts.
sce::SampleUtil::Graphics::VertexProgram	Vertex program.
sce::SampleUtil::Graphics::FragmentProgram	Fragment program.
sce::SampleUtil::Graphics::VertexStream	Vertex stream.
sce::SampleUtil::Graphics::VertexAttrib	Structure to indicate the layout of VertexBuffer .
sce::SampleUtil::Graphics::GraphicsLoader	Class to manage the resources required for drawing of graphics.
sce::SampleUtil::Graphics::Parameter	Parameter class of a shader.
sce::SampleUtil::Graphics::Program	ID of a shader program.

SCE CONFIDENTIAL

Item	Description
<u>sce::SampleUtil::Graphics::SpriteRenderer</u>	2D sprite rendering class
<u>sce::SampleUtil::Graphics::Object3dRenderer</u>	3D object renderer class

000004892117

Enumerated Types

MultisampleMode

Multisample mode.

Definition

```
#include <constant.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum MultisampleMode {
                kMultisampleNone,
                kMultisample2x,
                kMultisample4x
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kMultisampleNone	N/A	Single central sample.
kMultisample2x	N/A	2x diagonal samples
kMultisample4x	N/A	4x rotation grids

Description

This is the multisampling mode.

AttributeFormat

Format of a vertex attribute.

Definition

```
#include <constant.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum AttributeFormat {
                kAttributeFormatU8,
                kAttributeFormats8,
                kAttributeFormatU16,
                kAttributeFormats16,
                kAttributeFormatU8N,
                kAttributeFormats8N,
                kAttributeFormatU16N,
                kAttributeFormats16N,
                kAttributeFormatF16,
                kAttributeFormatF32
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kAttributeFormatU8	N/A	8-bit unsigned integer
kAttributeFormats8	N/A	8-bit signed integer
kAttributeFormatU16	N/A	16-bit unsigned integer
kAttributeFormats16	N/A	16-bit signed integer
kAttributeFormatU8N	N/A	8-bit unsigned integer normalized within the range of [0,1]
kAttributeFormats8N	N/A	8-bit signed integer normalized within the range of [-1,1]
kAttributeFormatU16N	N/A	16-bit unsigned integer normalized within the range of [0,1]
kAttributeFormats16N	N/A	16-bit signed integer normalized within the range of [-1,1]
kAttributeFormatF16	N/A	16-bit half-precision floating-point number
kAttributeFormatF32	N/A	32-bit single precision floating-point number

Description

This is the format of a vertex attribute.

ParameterSemantic

Semantic associated with a program parameter.

Definition

```
#include <constant.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum ParameterSemantic {
                kParameterSemanticNone,
                kParameterSemanticAttr,
                kParameterSemanticBcol,
                kParameterSemanticBinormal,
                kParameterSemanticBlendindices,
                kParameterSemanticBlendweight,
                kParameterSemanticColor,
                kParameterSemanticDiffuse,
                kParameterSemanticFogcoord,
                kParameterSemanticNormal,
                kParameterSemanticPointsize,
                kParameterSemanticPosition,
                kParameterSemanticSpecular,
                kParameterSemanticTangent,
                kParameterSemanticTexcoord
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kParameterSemanticNone	N/A	The parameter has no semantic.
kParameterSemanticAttr	N/A	The parameter has an ATTR semantic.
kParameterSemanticBcol	N/A	The parameter has a BCOL semantic.
kParameterSemanticBinormal	N/A	The parameter has a BINORMAL semantic.
kParameterSemanticBlendindices	N/A	The parameter has a BLENDINDICES semantic.
kParameterSemanticBlendweight	N/A	The parameter has a BLENDWEIGHT semantic.
kParameterSemanticColor	N/A	The parameter has a COLOR semantic.
kParameterSemanticDiffuse	N/A	The parameter has a DIFFUSE semantic.
kParameterSemanticFogcoord	N/A	The parameter has a FOGCOORD semantic.
kParameterSemanticNormal	N/A	The parameter has a NORMAL semantic.
kParameterSemanticPointsize	N/A	The parameter has a POINTSIZE semantic.
kParameterSemanticPosition	N/A	The parameter has a POSITION semantic.
kParameterSemanticSpecular	N/A	The parameter has a SPECULAR semantic.
kParameterSemanticTangent	N/A	The parameter has a TANGENT semantic.
kParameterSemanticTexcoord	N/A	The parameter has a TEXCOORD semantic.

Description

This is the semantic associated with a program parameter.

TextureAddrMode

The addressing mode of a texture.

Definition

```
#include <constant.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum TextureAddrMode {
                kTextureAddrModeRepeat,
                kTextureAddrModeMirror,
                kTextureAddrModeClamp,
                kTextureAddrModeMirrorClamp,
                kTextureAddrModeClampFullBorder
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kTextureAddrModeRepeat	N/A	This executes addressing of a texture using the fractional part of uv. In this case, the texture is repeated in this axis direction.
kTextureAddrModeMirror	N/A	f and (1-f) are used alternately. However, f is the fractional part of uv. In this case, the texture is repeated in this axis direction, but at that time, an inverted texture and a non-inverted one are repeated alternately.
kTextureAddrModeClamp	N/A	uv is clamped in the range of [0,1] to avoid filtering over the edge of a texture in the case that filtering is enabled. In this case, the result of "clamped at the edge" is generated.
kTextureAddrModeMirrorClamp	N/A	The absolute value of uv is clamped in the range of [0,1] to avoid filtering over the edge of a texture in the case that filtering is enabled. In this case, a texture mirrored once with 0 as the center is acquired.
kTextureAddrModeClampFullBorder	N/A	The border texel is used for the data outside [0,1]. Because the address is not clamped, if uv is outside [0,1], a border texel of up to 100% is acquired as a result of filtering.

Description

This is the addressing mode of a texture.

SCE CONFIDENTIAL

TextureFilter

[Texture](#) filter mode.

Definition

```
#include <constant.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum TextureFilter {
                kTextureFilterPoint,
                kTextureFilterLinear
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kTextureFilterPoint	N/A	Point sampling.
kTextureFilterLinear	N/A	Linear filtering.

Description

This is the texture filter mode.

SCE CONFIDENTIAL

TextureMipFilter

Mip map filter mode.

Definition

```
#include <constant.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum TextureMipFilter {
                kTextureMipFilterDisabled,
                kTextureMipFilterEnabled
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kTextureMipFilterDisabled	N/A	Filtering between mip maps is not executed.
kTextureMipFilterEnabled	N/A	Filtering between mip maps is executed.

Description

This is the mip map filter mode.

SCE CONFIDENTIAL

ProgramType

[Program](#) type.

Definition

```
#include <constant.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum ProgramType {
                kProgramTypeVertex,
                kProgramTypeFragment
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kProgramTypeVertex	N/A	The shader is a vertex program.
kProgramTypeFragment	N/A	The shader is a fragment program.

Description

This is a program type.

SCE CONFIDENTIAL

DepthFunc

Depth comparison function.

Definition

```
#include <constant.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum DepthFunc {
                kDepthFuncNever,
                kDepthFuncLess,
                kDepthFuncEqual,
                kDepthFuncLessEqual,
                kDepthFuncGreater,
                kDepthFuncNotEqual,
                kDepthFuncGreaterEqual,
                kDepthFuncAlways
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kDepthFuncNever	N/A	This does not pass.
kDepthFuncLess	N/A	Passes if the input depth is smaller than the saved depth.
kDepthFuncEqual	N/A	Passes if the input depth equals to the saved depth.
kDepthFuncLessEqual	N/A	Passes if the input depth equals to or smaller than the saved depth.
kDepthFuncGreater	N/A	Passes if the input depth is greater than the saved depth.
kDepthFuncNotEqual	N/A	Passes if the input depth is different from the saved depth.
kDepthFuncGreaterEqual	N/A	(1 - A_d, 1 - A_d, 1 - A_d, 1 - A_d)
kDepthFuncAlways	N/A	Always passes.

Description

This is the depth comparison function.

ParameterCategory

The category of a shader program parameter.

Definition

```
#include <constant.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum ParameterCategory {
                kParameterCategoryAttribute,
                kParameterCategoryUniform,
                kParameterCategorySampler,
                kParameterCategoryUnknown
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kParameterCategoryAttribute	N/A	The parameter is vertex attribute.
kParameterCategoryUniform	N/A	The parameter is uniform.
kParameterCategorySampler	N/A	The parameter is sampler.
kParameterCategoryUnknown	N/A	The parameter is uniform buffer.

Description

This is the category of a shader program parameter.

ParameterType

The datatype of a program parameter.

Definition

```
#include <constant.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum ParameterType {
                kParameterTypeF32,
                kParameterTypeF16,
                kParameterTypeC10,
                kParameterTypeU32,
                kParameterTypeS32,
                kParameterTypeU16,
                kParameterTypeS16,
                kParameterTypeU8,
                kParameterTypeS8,
                kParameterTypeSampler,
                kParameterTypeUnknown
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kParameterTypeF32	N/A	The 32-bit floating-point number corresponding to the "float" type of Cg.
kParameterTypeF16	N/A	The 16-bit floating-point number corresponding to the "half" type of Cg.
kParameterTypeC10	N/A	The 10-bit fixed-point number corresponding to the "fixed" type of Cg.
kParameterTypeU32	N/A	32-bit unsigned integer
kParameterTypeS32	N/A	32-bit signed integer
kParameterTypeU16	N/A	16-bit unsigned integer
kParameterTypeS16	N/A	16-bit signed integer
kParameterTypeU8	N/A	8-bit unsigned integer
kParameterTypeS8	N/A	8-bit signed integer
kParameterTypeSampler	N/A	Texture sampler.
kParameterTypeUnknown	N/A	Unknown type.

Description

This is the datatype of a program parameter.

SCE CONFIDENTIAL

CullMode

Back-face culling mode.

Definition

```
#include <constant.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum CullMode {
                kCullNone,
                kCullCw,
                kCullCcw
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kCullNone	N/A	No culling.
kCullCw	N/A	Culls a triangle of clockwise rotation in the window coordinate.
kCullCcw	N/A	Culls a triangle of counterclockwise rotation in the window coordinate.

Description

This is the back-face culling mode.

SCE CONFIDENTIAL

Primitive

Primitive type.

Definition

```
#include <constant.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum Primitive {
                kPrimitivePoints,
                kPrimitiveLines,
                kPrimitiveTriangles,
                kPrimitiveTriangleStrip
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kPrimitivePoints	N/A	Indexed point list.
kPrimitiveLines	N/A	Indexed line list.
kPrimitiveTriangles	N/A	Indexed triangle list.
kPrimitiveTriangleStrip	N/A	Indexed triangle strip.

Description

This is a primitive type.

IndexSource

The type of an index source for indexing of a vertex stream.

Definition

```
#include <constant.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum IndexSource {
                kIndexSourceIndex16Bit,
                kIndexSourceIndex32Bit,
                kIndexSourceInstance16Bit,
                kIndexSourceInstance32Bit
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kIndexSourceIndex16Bit	N/A	The type of an index stream. The stream is indexed using index values. All values have to be lower than 64K. However, the U16 and U32 formats can also be used in memory.
kIndexSourceIndex32Bit	N/A	The stream is indexed using index values.
kIndexSourceInstance16Bit	N/A	The stream is indexed using instance numbers. This can only be used for a drawing call including 64 K or less instances.
kIndexSourceInstance32Bit	N/A	The stream is indexed using instance numbers.

Description

This is the type of an index source for indexing of vertex stream.

SCE CONFIDENTIAL

StencilFunc

Stencil comparison function.

Definition

```
#include <constant.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum StencilFunc {
                kStencilFuncNever,
                kStencilFuncLess,
                kStencilFuncEqual,
                kStencilFuncLessEqual,
                kStencilFuncGreater,
                kStencilFuncNotEqual,
                kStencilFuncGreaterEqual,
                kStencilFuncAlways
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kStencilFuncNever	N/A	This does not pass.
kStencilFuncLess	N/A	Passes if (reference & mask) is smaller than (stencil & mask).
kStencilFuncEqual	N/A	Passes if (reference & mask) equals to (stencil & mask).
kStencilFuncLessEqual	N/A	Passes if (reference & mask) equals to or smaller than (stencil & mask).
kStencilFuncGreater	N/A	Passes if (reference & mask) is greater than (stencil & mask).
kStencilFuncNotEqual	N/A	Passes if (reference & mask) does not equal to (stencil & mask).
kStencilFuncGreaterEqual	N/A	Passes if (reference & mask) equals to or greater than (stencil & mask).
kStencilFuncAlways	N/A	Always passes.

Description

This is the stencil comparison function.

StencilOp

Stencil operation.

Definition

```
#include <constant.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum StencilOp {
                kStencilOpKeep,
                kStencilOpZero,
                kStencilOpReplace,
                kStencilOpIncr,
                kStencilOpDecr,
                kStencilOpInvert,
                kStencilOpIncrWrap,
                kStencilOpDecrWrap
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kStencilOpKeep	N/A	Retains the current stencil buffer value.
kStencilOpZero	N/A	Sets the stencil buffer value to 0.
kStencilOpReplace	N/A	Replaces the value of a stencil buffer with a stencil reference value.
kStencilOpIncr	N/A	Increments the value of a stencil buffer. (Clamped at 255 if it exceeds 255)
kStencilOpDecr	N/A	Decrements the value of a stencil buffer. (Clamped at 0 if it exceeds 0)
kStencilOpInvert	N/A	Bit-inverts the value of a stencil buffer.
kStencilOpIncrWrap	N/A	Increments the value of a stencil buffer. (Lapped at 0 if it exceeds 255)
kStencilOpDecrWrap	N/A	Decrements the value of a stencil buffer. (Lapped at 255 if it exceeds 0)

Description

This is a stencil operation.

SCE CONFIDENTIAL

TextureFormat

[Texture](#) format.

Definition

```
#include <constant.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum TextureFormat {
                kTextureFormatA8B8G8R8Unorm,
                kTextureFormatR8Unorm
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kTextureFormatA8B8G8R8Unorm	N/A	The U8U8U8U8 data is read from memory in the order of ABGR.
kTextureFormatR8Unorm	N/A	The U8 data is read from memory as R.

Description

This is the texture format.

SCE CONFIDENTIAL

ColorMask

Color mask for shader code patching of a runtime by the shader patcher.

Definition

```
#include <constant.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum ColorMask {
                kColorMaskNone = 0,
                kColorMaskA = (1 << 0),
                kColorMaskR = (1 << 1),
                kColorMaskG = (1 << 2),
                kColorMaskB = (1 << 3),
                kColorMaskAll = (kColorMaskA | kColorMaskB |
                    kColorMaskG | kColorMaskR)
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kColorMaskNone	0	Not written in any channels.
kColorMaskA	(1 << 0)	Written in the alpha channel.
kColorMaskR	(1 << 1)	Written in the red channel.
kColorMaskG	(1 << 2)	Written in the green channel.
kColorMaskB	(1 << 3)	Written in the blue channel.
kColorMaskAll	(kColorMaskA kColorMaskB kColorMaskG kColorMaskR)	Written in all channels.

Description

This is the color mask for shader code patching of a runtime by the shader patcher.

SCE CONFIDENTIAL

BlendFunc

Blend function for shader code patching of a runtime by the shader patcher.

Definition

```
#include <constant.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum BlendFunc {
                kBlendFuncNone,
                kBlendFuncAdd,
                kBlendFuncSubtract,
                kBlendFuncReverseSubtract,
                kBlendFuncMin,
                kBlendFuncMax
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kBlendFuncNone	N/A	$D = S.$
kBlendFuncAdd	N/A	$D = S * Coefficient S + D * Coefficient D.$
kBlendFuncSubtract	N/A	$D = S * Coefficient S - D * Coefficient D.$
kBlendFuncReverseSubtract	N/A	$D = D * Coefficient D - S * Coefficient S.$
kBlendFuncMin	N/A	$D = \min(S, D)$
kBlendFuncMax	N/A	$D = \max(S, D)$

Description

This is the blend function for shader code patching of a runtime by the shader patcher.

BlendFactor

The blend coefficient for shader code patching of a runtime by the shader patcher.

Definition

```
#include <constant.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum BlendFactor {
                kBlendFactorZero,
                kBlendFactorOne,
                kBlendFactorSrcColor,
                kBlendFactorOneMinusSrcColor,
                kBlendFactorSrcAlpha,
                kBlendFactorOneMinusSrcAlpha,
                kBlendFactorDstColor,
                kBlendFactorOneMinusDstColor,
                kBlendFactorDstAlpha,
                kBlendFactorOneMinusDstAlpha
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kBlendFactorZero	N/A	(0, 0, 0, 0)
kBlendFactorOne	N/A	(1, 1, 1, 1)
kBlendFactorSrcColor	N/A	(R_s, G_s, B_s, A_s)
kBlendFactorOneMinusSrcColor	N/A	(1 - R_s, 1 - G_s, 1 - B_s, 1 - A_s)
kBlendFactorSrcAlpha	N/A	(A_s, A_s, A_s, A_s)
kBlendFactorOneMinusSrcAlpha	N/A	(1 - A_s, 1 - A_s, 1 - A_s, 1 - A_s)
kBlendFactorDstColor	N/A	(R_d, G_d, B_d, A_d)
kBlendFactorOneMinusDstColor	N/A	(1 - R_d, 1 - G_d, 1 - B_d, 1 - A_d)
kBlendFactorDstAlpha	N/A	(A_d, A_d, A_d, A_d)
kBlendFactorOneMinusDstAlpha	N/A	(1 - A_d, 1 - A_d, 1 - A_d, 1 - A_d)

Description

This is the blend coefficient for shader code patching of a runtime by the shader patcher.

SCE CONFIDENTIAL

ShaderFormat

Shader format.

Definition

```
#include <constant.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum ShaderFormat {
                kShaderFormatGxp,
                kShaderFormatDefault
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kShaderFormatGxp	N/A	PlayStation®Vita GXP format (can only be used on PlayStation®Vita)
kShaderFormatDefault	N/A	Default format on each platform. PlayStation®4 is kShaderFormatPsslBinary, PlayStation®Vita is kShaderFormatGxp, Windows is kShaderFormatCgSource.

Description

This enumerated type represents the shader format. It is passed together with shader data upon creating a shader object.

BufferBindFlag

[Buffer](#) flag.

Definition

```
#include <constant.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum BufferBindFlag {
                kBufferBindFlagShaderResource = 1<<0,
                kBufferBindFlagRenderTarget = 1<<1,
                kBufferBindFlagDepthStencil = 1<<2,
                kBufferBindFlagVertexBuffer = 1<<3,
                kBufferBindFlagIndexBuffer = 1<<4,
                kBufferBindFlagConstantBuffer = 1<<5,
                kBufferBindFlagUnorderedAccess = 1<<6
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kBufferBindFlagShaderResource	1<<0	Buffer used as a shader resource such as, a texture buffer or structured buffer.
kBufferBindFlagRenderTarget	1<<1	Buffer used as a render target.
kBufferBindFlagDepthStencil	1<<2	Buffer used as a depth stencil target.
kBufferBindFlagVertexBuffer	1<<3	Buffer used as a vertex buffer.
kBufferBindFlagIndexBuffer	1<<4	Buffer used as an index buffer.
kBufferBindFlagConstantBuffer	1<<5	Buffer used as a constant buffer.
kBufferBindFlagUnorderedAccess	1<<6	Buffer used as an unordered access buffer.

Description

This enumerated type represents flags to indicate how a buffer is to be used in the graphics pipeline

SCE CONFIDENTIAL

BufferAccessMode

Buffer access mode.

Definition

```
#include <constant.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum BufferAccessMode {
                kBufferAccessModeGpuReadWrite,
                kBufferAccessModeGpuReadOnly,
                kBufferAccessModeGpuReadCpuWrite
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kBufferAccessModeGpuReadWrite	N/A	Read/write from the GPU enabled.
kBufferAccessModeGpuReadOnly	N/A	Only read from the GPU enabled.
kBufferAccessModeGpuReadCpuWrite	N/A	Read from the GPU and write from the CPU enabled.

Description

This enumerated type represents the buffer access mode.

BufferFormat

[Buffer](#) element format.

Definition

```
#include <constant.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum BufferFormat {
                kBufferFormatUnknown,
                kBufferFormatX8X8X8X8,
                kBufferFormatX8X8,
                kBufferFormatX8,
                kBufferFormatX16X16X16X16,
                kBufferFormatX16X16,
                kBufferFormatX16,
                kBufferFormatX32X8X24,
                kBufferFormatX32,
                kBufferFormatX24X8
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kBufferFormatUnknown	N/A	Unknown format.
kBufferFormatX8X8X8X8	N/A	Three 8-bit values.
kBufferFormatX8X8	N/A	Two 8-bit values.
kBufferFormatX8	N/A	One 8-bit value.
kBufferFormatX16X16X16X16	N/A	Four 16-bit values.
kBufferFormatX16X16	N/A	Two 16-bit values.
kBufferFormatX16	N/A	One 16-bit value.
kBufferFormatX32X8X24	N/A	One 32-bit value, one 8-bit value, and one 24-bit value.
kBufferFormatX32	N/A	One 32-bit value.
kBufferFormatX24X8	N/A	One 24-bit value and one 8-bit value.

Description

This enumerated type represents the buffer element format.

SCE CONFIDENTIAL

BufferDimension

[Buffer](#) dimension.

Definition

```
#include <constant.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum BufferDimension {
                kBufferDimensionBuffer,
                kBufferDimensionTexture1D,
                kBufferDimensionTexture2D,
                kBufferDimensionTexture3D
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kBufferDimensionBuffer	N/A	One-dimensional number.
kBufferDimensionTexture1D	N/A	One-dimensional texture.
kBufferDimensionTexture2D	N/A	Two-dimensional texture.
kBufferDimensionTexture3D	N/A	Three-dimensional texture.

Description

This enumerated type represents the buffer dimension.

SCE CONFIDENTIAL

FontLanguage

Photo language code.

Definition

```
#include <font.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum FontLanguage {
                kFontLanguageLatin,
                kFontLanguageJapanese,
                kFontLanguageChinese,
                kFontLanguageKorean
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kFontLanguageLatin	N/A	English.
kFontLanguageJapanese	N/A	Japanese.
kFontLanguageChinese	N/A	Chinese.
kFontLanguageKorean	N/A	Korean.

Description

This is the photo language code.

SCE CONFIDENTIAL

FontWeight

[Font](#) weight.

Definition

```
#include <font.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum FontWeight {
                kFontWeightNarrow,
                kFontWeightNormal,
                kFontWeightBold,
                kFontWeightBlack
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kFontWeightNarrow	N/A	Narrow.
kFontWeightNormal	N/A	Normal thickness.
kFontWeightBold	N/A	Bold.
kFontWeightBlack	N/A	Extra bold.

Description

This value indicates the font thickness.

SCE CONFIDENTIAL

FontFamily

[Font](#) family.

Definition

```
#include <font.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            enum FontFamily {
                kFontFamilySansSerif,
                kFontFamilySerif,
                kFontFamilyRounded
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kFontFamilySansSerif	N/A	Sans serif.
kFontFamilySerif	N/A	Serif.
kFontFamilyRounded	N/A	Rounded.

Description

This is the font family. It specifies the font type.

Functions

createGraphicsContext

Generate graphics context.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            int createGraphicsContext(
                GraphicsContext **outGraphicsContext,
                uint32_t width,
                uint32_t height,
                GraphicsContextOption *option = NULL
            );
        }
    }
}
```

Arguments

<i>outGraphicsContext</i>	Pointer to which the generated graphics context returns
<i>width</i>	Width of the main render target displayed on the display
<i>height</i>	Height of the main render target displayed on the display
<i>option</i>	GraphicsContextOption structure. This is initialized by the default value if NULL is specified.

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This generates a graphics context. To delete the generated instance, use [sce::SampleUtil::destroy\(\)](#).

SCE CONFIDENTIAL

createFontLoader

Create [FontLoader](#) class.

Definition

```
#include <font.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            int createFontLoader(
                FontLoader **outFontLoader,
                GraphicsLoader *loader
            );
        }
    }
}
```

Arguments

<i>outFontLoader</i>	Pointer to the FontLoader generated
<i>loader</i>	Pointer to GraphicsLoader

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Failure

Description

This generates the [FontLoader](#) class.

SCE CONFIDENTIAL

createSpriteRenderer

Create [SpriteRenderer](#).

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            int createSpriteRenderer(
                SpriteRenderer **outSpriteRenderer,
                GraphicsLoader *loader
            );
        }
    }
}
```

Arguments

<i>outSpriteRenderer</i>	Pointer to which the generated SpriteRenderer returns
<i>loader</i>	GraphicsLoader instance

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid

Description

This generates the [SpriteRenderer](#). To delete the generated [SpriteRenderer](#), use [sce::SampleUtil::destroy\(\)](#).

SCE CONFIDENTIAL

createObject3dRenderer

Generate [Object3dRenderer](#).

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            int createObject3dRenderer(
                Object3dRenderer **outObject3dRenderer,
                GraphicsLoader *loader
            );
        }
    }
}
```

Arguments

<i>outObject3dRenderer</i>	Pointer to which the generated Object3dRenderer returns.
<i>loader</i>	GraphicsLoader instance

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This generates the [Object3dRenderer](#). To delete the generated [Object3dRenderer](#), use [sce::SampleUtil::destroy\(\)](#).

sce::SampleUtil::Graphics::BufferInterface

ce

000004892117

Summary

sce::SampleUtil::Graphics::BufferInterface

Class which becomes the base of a buffer object.

Definition

```
#include <buffer.h>
class BufferInterface {};
```

Description

This is the class which becomes the base of a buffer object.

Methods Summary

Methods	Description
beginWrite	Starts writing to the buffer.
endWrite	Terminates writing to the buffer.
getLoader	Gets the pointer to GraphicsLoader .
getDimension	Return buffer dimension.

Public Instance Methods

beginWrite

Starts writing to the buffer.

Definition

```
#include <buffer.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class BufferInterface {
                virtual void *beginWrite(void)=0;
            }
        }
    }
}
```

Return Values

Pointer to the data area of a buffer object

Description

This starts writing to the buffer.

SCE CONFIDENTIAL

endWrite

Terminates writing to the buffer.

Definition

```
#include <buffer.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class BufferInterface {
                virtual void endWrite(void)=0;
            }
        }
    }
}
```

Return Values

None

Description

This terminates writing to the buffer.

SCE CONFIDENTIAL

getLoader

Gets the pointer to [GraphicsLoader](#).

Definition

```
#include <buffer.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class BufferInterface {
                virtual GraphicsLoader *getLoader(void)=0 const;
            }
        }
    }
}
```

Return Values

Pointer to [GraphicsLoader](#)

Description

This gets the pointer to [GraphicsLoader](#).

SCE CONFIDENTIAL

getDimension

Return buffer dimension.

Definition

```
#include <buffer.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class BufferInterface {
                virtual BufferDimension getDimension(void)=0
            const;
        }
    }
}
```

Return Values

[Buffer](#) dimension

Description

This function returns the buffer dimension.

sce::SampleUtil::Graphics::Buffer

Summary

sce::SampleUtil::Graphics::Buffer

One-dimensional buffer class.

Definition

```
#include <buffer.h>
class Buffer : public sce::SampleUtil::Resource,
public sce::SampleUtil::Graphics::BufferInterface {};
```

Description

A low-level buffer implementation of a one-dimensional buffer object such as, [IndexBuffer](#) and [StructuredBuffer](#).

Methods Summary

Methods	Description
beginWrite	Starts writing to the buffer.
endWrite	Terminates writing to the buffer.
getLoader	Gets the pointer to GraphicsLoader .
getSize	Returns the data area size of a buffer object.
getBindFlags	Returns the bind flag of a buffer object.
getAccessMode	Returns the access flag of a buffer object.
getElementSize	Returns the size of one buffer object element.

Public Instance Methods

beginWrite

Starts writing to the buffer.

Definition

```
#include <buffer.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Buffer {
                virtual void *beginWrite(void)=0;
            }
        }
    }
}
```

Return Values

Pointer to the data area of a buffer object

Description

This starts writing to the buffer.

SCE CONFIDENTIAL

endWrite

Terminates writing to the buffer.

Definition

```
#include <buffer.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Buffer {
                virtual void endWrite(void)=0;
            }
        }
    }
}
```

Return Values

None

Description

This terminates writing to the buffer.

SCE CONFIDENTIAL

getLoader

Gets the pointer to [GraphicsLoader](#).

Definition

```
#include <buffer.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Buffer {
                virtual GraphicsLoader *getLoader(void)=0 const;
            }
        }
    }
}
```

Return Values

Pointer to [GraphicsLoader](#)

Description

This gets the pointer to [GraphicsLoader](#).

SCE CONFIDENTIAL

getSize

Returns the data area size of a buffer object.

Definition

```
#include <buffer.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Buffer {
                virtual size_t getSize(void)=0 const;
            }
        }
    }
}
```

Return Values

Data area size of a buffer object

Description

This returns the data area size of a buffer object.

SCE CONFIDENTIAL

getBindFlags

Returns the bind flag of a buffer object.

Definition

```
#include <buffer.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Buffer {
                virtual uint32_t getBindFlags(void)=0 const;
            }
        }
    }
}
```

Return Values

Bind flag of a buffer object

Description

This returns the bind flag of a buffer object. The bind flag is a bit OR of kBufferBindFlag....

SCE CONFIDENTIAL

getAccessMode

Returns the access flag of a buffer object.

Definition

```
#include <buffer.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Buffer {
                virtual BufferAccessMode getAccessMode(void)=0
            const;
        }
    }
}
```

Return Values

Access flag of a buffer object

Description

This returns the access flag of a buffer object.

SCE CONFIDENTIAL

getElementsSize

Returns the size of one buffer object element.

Definition

```
#include <buffer.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Buffer {
                virtual size_t getElementsSize(void)=0 const;
            }
        }
    }
}
```

Return Values

Size of one buffer object element

Description

This returns the size of one buffer object element.

sce::SampleUtil::Graphics::IndexBuffer

000004892117

Summary

sce::SampleUtil::Graphics::IndexBuffer

Index buffer.

Definition

```
#include <buffer.h>
class IndexBuffer : public sce::SampleUtil::Resource,
public sce::SampleUtil::Graphics::BufferInterface {};
```

Description

This is the index buffer class.

Methods Summary

Methods	Description
getBaseBuffer	Returns a low-level buffer object.

Protected Instance Methods

getBaseBuffer

Returns a low-level buffer object.

Definition

```
#include <buffer.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class IndexBuffer {
                virtual Buffer *getBaseBuffer(void)=0;
            }
        }
    }
}
```

Return Values

Low-level buffer object

Description

This returns a low-level buffer object.

sce::SampleUtil::Graphics::VertexBuffer

000004892117

Summary

sce::SampleUtil::Graphics::VertexBuffer

Vertex buffer.

Definition

```
#include <buffer.h>
class VertexBuffer : public sce::SampleUtil::Resource,
public sce::SampleUtil::Graphics::BufferInterface {};
```

Description

This is the vertex buffer class.

Methods Summary

Methods	Description
getBaseBuffer	Returns a low-level buffer object.

Protected Instance Methods

getBaseBuffer

Returns a low-level buffer object.

Definition

```
#include <buffer.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class VertexBuffer {
                virtual Buffer *getBaseBuffer(void)=0;
            }
        }
    }
}
```

Return Values

Low-level buffer object

Description

This returns a low-level buffer object.

sce::SampleUtil::Graphics::UniformBuff

er

000004892117

Summary

sce::SampleUtil::Graphics::UniformBuffer

Uniform buffer.

Definition

```
#include <buffer.h>
class UniformBuffer : public sce::SampleUtil::Resource,
public sce::SampleUtil::Graphics::BufferInterface {};
```

Description

This is the uniform buffer class.

Methods Summary

Methods	Description
getBaseBuffer	Returns a low-level buffer object.

Protected Instance Methods

getBaseBuffer

Returns a low-level buffer object.

Definition

```
#include <buffer.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class UniformBuffer {
                virtual Buffer *getBaseBuffer(void)=0;
            }
        }
    }
}
```

Return Values

Low-level buffer object

Description

This returns a low-level buffer object.

sce::SampleUtil::Graphics::StructuredBuffer

Summary

sce::SampleUtil::Graphics::StructuredBuffer

Structured buffer.

Definition

```
#include <buffer.h>
class StructuredBuffer : public sce::SampleUtil::Resource,
public sce::SampleUtil::Graphics::BufferInterface {};
```

Description

Structuring.

Methods Summary

Methods	Description
getBaseBuffer	Returns a low-level buffer object.

Protected Instance Methods

getBaseBuffer

Returns a low-level buffer object.

Definition

```
#include <buffer.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class StructuredBuffer {
                virtual Buffer *getBaseBuffer(void)=0;
            }
        }
    }
}
```

Return Values

Low-level buffer object

Description

This returns a low-level buffer object.

sce::SampleUtil::Graphics::Texture

000004892117

Summary

sce::SampleUtil::Graphics::Texture

[Texture](#) buffer class.

Definition

```
#include <buffer.h>
class Texture : public sce::SampleUtil::Resource,
public sce::SampleUtil::Graphics::BufferInterface {};
```

Description

This is the texture buffer class.

Methods Summary

Methods	Description
setAddrMode	Sets the addressing mode of a texture.
setFilter	Sets a texture filter.
getWidth	Gets the width of a texture.
getHeight	Gets the height of a texture.
getStride	Gets the stride of a texture.

Public Instance Methods

setAddrMode

Sets the addressing mode of a texture.

Definition

```
#include <buffer.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Texture {
                virtual int setAddrMode(
                    TextureAddrMode uAddrMode,
                    TextureAddrMode vAddrMode
                )=0;
            }
        }
    }
}
```

Arguments

uAddrMode U addressing mode
vAddrMode V addressing mode

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This sets the addressing mode of a texture.

SCE CONFIDENTIAL

setFilter

Sets a texture filter.

Definition

```
#include <buffer.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Texture {
                virtual int setFilter(
                    TextureFilter minFilter,
                    TextureFilter magFilter,
                    TextureMipFilter mipFilter
                )=0;
            }
        }
    }
}
```

Arguments

minFilter Reduction filter mode
magFilter Magnification filter mode
mipFilter Mip filter mode

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This sets a texture filter.

SCE CONFIDENTIAL

getWidth

Gets the width of a texture.

Definition

```
#include <buffer.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Texture {
                virtual uint32_t getWidth(void)=0 const;
            }
        }
    }
}
```

Return Values

Width of a texture

Description

This gets the width of a texture.

SCE CONFIDENTIAL

getHeight

Gets the height of a texture.

Definition

```
#include <buffer.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Texture {
                virtual uint32_t getHeight(void)=0 const;
            }
        }
    }
}
```

Return Values

Height of a texture

Description

This gets the height of a texture.

SCE CONFIDENTIAL

getStride

Gets the stride of a texture.

Definition

```
#include <buffer.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Texture {
                virtual uint32_t getStride(void)=0 const;
            }
        }
    }
}
```

Return Values

Stride of a texture (Unit: byte)

Description

This gets the stride of a texture. For type other than SCE_GXM_TEXTURE_LINEAR_STRIDED, 0 is returned.

sce::SampleUtil::Graphics::RenderTarget

et

000004892117

Summary

sce::SampleUtil::Graphics::RenderTarget

Render target.

Definition

```
#include <buffer.h>
class RenderTarget : public sce::SampleUtil::Resource {};
```

Description

This is a render target.

Methods Summary

Methods	Description
getWidth	Gets the width of a render target.
getHeight	Gets the height of a render target.
getTexture	Gets the pointer to a GXM texture.

Public Instance Methods

getWidth

Gets the width of a render target.

Definition

```
#include <buffer.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class RenderTarget {
                virtual uint32_t getWidth(void)=0 const;
            }
        }
    }
}
```

Return Values

Width of the render target

Description

This gets the width of a render target.

SCE CONFIDENTIAL

getHeight

Gets the height of a render target.

Definition

```
#include <buffer.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class RenderTarget {
                virtual uint32_t getHeight(void)=0 const;
            }
        }
    }
}
```

Return Values

Height of the render target

Description

This gets the height of a render target

SCE CONFIDENTIAL

getTexture

Gets the pointer to a GXM texture.

Definition

```
#include <buffer.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class RenderTarget {
                virtual Texture *getTexture(void)=0;
            }
        }
    }
}
```

Return Values

Pointer to a GXM texture

Description

This gets the pointer to a GXM texture.

sce::SampleUtil::Graphics::DepthStencilSurface

Summary

sce::SampleUtil::Graphics::DepthStencilSurface

Depth stencil target.

Definition

```
#include <buffer.h>
class DepthStencilSurface : public sce::SampleUtil::Resource {};
```

Description

This is a depth stencil target.

Methods Summary

Methods	Description
getTexture	Gets the pointer to a depth and stencil texture.

Public Instance Methods

getTexture

Gets the pointer to a depth and stencil texture.

Definition

```
#include <buffer.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class DepthStencilSurface {
                virtual Texture *getTexture(void)=0;
            }
        }
    }
}
```

Return Values

Pointer to a depth and stencil texture

Description

This gets the pointer to a depth and stencil texture.

sce::SampleUtil::Graphics::Collada::ColladaData

Summary

sce::SampleUtil::Graphics::Collada::ColladaData

Class of the Collada data.

Definition

```
#include <collada.h>
class ColladaData : public sce::SampleUtil::Resource {};
```

Description

This is the class to handle the Collada data.

Methods Summary

Methods	Description
getLoader	Returns ColladaLoader used for initialization.
getImageById	Get image.
getImageByIndex	Get image.
getImageByIndex	Get image.
getNumImages	Get number of images.
getMaterialById	Get material.
getMaterialByIndex	Get material.
getNumMaterials	Get number of materials.
getMeshByControllerId	Get mesh by specifying the controller ID.
getMeshByGeometryId	Gets a mesh by specifying a geometry ID.
getMeshByIndex	Gets a mesh by specifying an index.
getNumMeshes	Gets the number of meshes.
getVisualScene	Gets a scene.
getAnimation	Gets animation data.

Public Instance Methods

getLoader

Returns [ColladaLoader](#) used for initialization.

Definition

```
#include <collada.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class ColladaData {
                    virtual ColladaLoader *getLoader ()=0
                const;
            }
        }
    }
}
```

Return Values

Pointer to [ColladaLoader](#)

Description

This returns [ColladaLoader](#) used for initialization.

SCE CONFIDENTIAL

getImageById

Get image.

Definition

```
#include <collada.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class ColladaData {
                    virtual const Image *getImageById(
                        std::string id
                    )=0 const;
                }
            }
        }
    }
}
```

Arguments

id [Image](#) ID

Return Values

Pointer to [Image](#). NULL if not found.

Description

This gets an image.

SCE CONFIDENTIAL

getImageByIndex

Get image.

Definition

```
#include <collada.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class ColladaData {
                    virtual const Image *getImageByIndex(
                        uint32_t index
                    )=0 const;
                }
            }
        }
    }
}
```

Arguments

index Index of images. 0 or higher and lower than [getNumImages \(\)](#).

Return Values

Pointer to [Image](#). NULL if not found.

Description

This gets an image.

SCE CONFIDENTIAL

getImageByIndex

Get image.

Definition

```
#include <collada.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class ColladaData {
                    virtual Image *getImageByIndex(
                        uint32_t index
                    )=0;
                }
            }
        }
    }
}
```

Arguments

index Index of images. 0 or higher and lower than [getNumImages \(\)](#).

Return Values

Pointer to [Image](#). NULL if not found.

Description

This gets an image.

SCE CONFIDENTIAL

getNumImages

Get number of images.

Definition

```
#include <collada.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class ColladaData {
                    virtual uint32_t getNumImages(void)=0;
                }
            }
        }
    }
}
```

Return Values

Number of images

Description

Gets the number of images.

SCE CONFIDENTIAL

getMaterialById

Get material.

Definition

```
#include <collada.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class ColladaData {
                    virtual const Material *getMaterialById(
                        std::string id
                    )=0 const;
                }
            }
        }
    }
}
```

Arguments

id ID of a material

Return Values

Pointer to [Material](#). NULL if not found.

Description

This gets a material.

SCE CONFIDENTIAL

getMaterialByIndex

Get material.

Definition

```
#include <collada.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class ColladaData {
                    virtual const Material*
*getMaterialByIndex(
                        uint32_t index
                    )=0 const;
                };
            }
        }
    }
}
```

Arguments

index Index of materials. 0 or higher and lower than [getNumMaterials\(\)](#).

Return Values

Pointer to [Material](#). NULL if not found.

Description

This gets a material.

SCE CONFIDENTIAL

getNumMaterials

Get number of materials.

Definition

```
#include <collada.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class ColladaData {
                    virtual uint32_t getNumMaterials(void)=0
                const;
            }
        }
    }
}
```

Return Values

Number of materials

Description

This gets the number of materials.

SCE CONFIDENTIAL

getMeshByControllerId

Get mesh by specifying the controller ID.

Definition

```
#include <collada.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class ColladaData {
                    virtual const Mesh *getMeshByControllerId(
                        std::string controllerId
                    )=0 const;
                }
            }
        }
    }
}
```

Arguments

controllerId [Mesh](#) controller ID

Return Values

Pointer to [Mesh](#). NULL if not found.

Description

This function obtains the mesh for the specified controller ID.

getMeshByGeometryId

Gets a mesh by specifying a geometry ID.

Definition

```
#include <collada.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class ColladaData {
                    virtual const Mesh *getMeshByGeometryId(
                        std::string geometryId
                    )=0 const;
                }
            }
        }
    }
}
```

Arguments

geometryId Geometry ID

Return Values

Pointer to [Mesh](#). NULL if not found.

Description

This gets a mesh by specifying a geometry ID.

SCE CONFIDENTIAL

getMeshByIndex

Gets a mesh by specifying an index.

Definition

```
#include <collada.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class ColladaData {
                    virtual const Mesh *getMeshByIndex(
                        uint32_t index
                    )=0 const;
                }
            }
        }
    }
}
```

Arguments

index Index of meshes. 0 or higher and lower than [getNumMeshes\(\)](#).

Return Values

Pointer to [Mesh](#). NULL if not found.

Description

This gets a mesh by specifying an index.

SCE CONFIDENTIAL

getNumMeshes

Gets the number of meshes.

Definition

```
#include <collada.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class ColladaData {
                    virtual uint32_t getNumMeshes(void)=0
                const;
            }
        }
    }
}
```

Return Values

Number of meshes

Description

This gets the number of meshes

SCE CONFIDENTIAL

getVisualScene

Gets a scene.

Definition

```
#include <collada.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class ColladaData {
                    virtual const VisualScene
*getVisualScene(void)=0 const;
                }
            }
        }
    }
}
```

Return Values

Pointer to [VisualScene](#). NULL if not found.

Description

This gets a scene.

SCE CONFIDENTIAL

getAnimation

Gets animation data.

Definition

```
#include <collada.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class ColladaData {
                    virtual const Animation*
*getAnimation(void)=0 const;
                }
            }
        }
    }
}
```

Return Values

Pointer to [Animation](#). NULL if not found.

Description

This gets animation data.

sce::SampleUtil::Graphics::Collada::NodeTransform

Summary

sce::SampleUtil::Graphics::Collada::NodeTransform

Transform of a node in a scene.

Definition

```
#include <collada_animation.h>
struct NodeTransform {};
```

Description

This is the transform of a node in a scene.

Fields

Public Instance Fields

sce::Vectormath::Simd::Aos::Vector3 <i>scale</i>	Scaling element of transform.
sce::Vectormath::Simd::Aos::Quat <i>rotation</i>	Rotation element of transform.
sce::Vectormath::Simd::Aos::Vector3 <i>translation</i>	Parallel displacement element of transform.

sce::SampleUtil::Graphics::Collada::No deAnimation

Summary

sce::SampleUtil::Graphics::Collada::NodeAnimation

[Animation](#) data of one node.

Definition

```
#include <collada_animation.h>
class NodeAnimation {};
```

Description

This is the animation data of one node.

Methods Summary

Methods	Description
getNodeId	Returns ID of target node of animation.
getNumFrames	Returns number of frames of animation.
getNodeTransform	Returns transform of specified frame.
getMatrix	Return matrix of specified frame.

Public Instance Methods

getNodeId

Returns ID of target node of animation.

Definition

```
#include <collada_animation.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class NodeAnimation {
                    std::string nodeId(void) const;
                }
            }
        }
    }
}
```

Return Values

ID of the target node of animation

Description

This returns the ID of the target node of animation.

SCE CONFIDENTIAL

getNumFrames

Returns number of frames of animation.

Definition

```
#include <collada_animation.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class NodeAnimation {
                    uint32_t getNumFrames(void) const;
                }
            }
        }
    }
}
```

Return Values

Number of frames of animation

Description

This returns the number of frames of animation.

SCE CONFIDENTIAL

getNodeTransform

Returns transform of specified frame.

Definition

```
#include <collada_animation.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class NodeAnimation {
                    NodeTransform getNodeTransform(
                        uint32_t frameIndex
                    ) const;
                }
            }
        }
    }
}
```

Arguments

frameIndex Frame number

Return Values

Transform of the specified frame

Description

This returns the transform of the specified frame.

SCE CONFIDENTIAL

getMatrix

Return matrix of specified frame.

Definition

```
#include <collada_animation.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class NodeAnimation {
                    sce::Vectormath::Simd::Aos::Matrix4
getMatrix(
                    uint32_t frameIndex
                ) const;
            }
        }
    }
}
```

Arguments

frameIndex Frame number

Return Values

The matrix of the specified frame

Description

This returns the matrix of the specified frame.

sce::SampleUtil::Graphics::Collada::ShaderParameterReference

Summary

sce::SampleUtil::Graphics::Collada::ShaderParameterReference

The interface for a class to refer to a shader parameter.

Definition

```
#include <collada_base.h>
class ShaderParameterReference { };
```

Description

This is the interface to manage the reference to a shader parameter value. This is used when managing the values of parameters shared in a mesh in a scene in bulk.

Specify the instance of the class that inherited this interface in Node::applyParameterReference(). For the parameters to which a value other than NULL is returned when [getParamValue\(\)](#) is called to all parameters, the values will be referenced every time rendering is executed.

Methods Summary

Methods	Description
~ShaderParameterReference	Destructor.
getParamValue	Returns the reference destination of a shader parameter.

Constructors and Destructors

~ShaderParameterReference

Destructor.

Definition

```
#include <collada_base.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class ShaderParameterReference {
                    virtual inline
~ShaderParameterReference(void);
                };
            }
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Public Instance Methods

getParamValue

Returns the reference destination of a shader parameter.

Definition

```
#include <collada_base.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class ShaderParameterReference {
                    virtual Graphics::Effect::ParameterValue
*getParamValue (
                    Graphics::ProgramType programType,
                    const Graphics::Parameter
*parameter
                )=0;
            }
        }
    }
}
```

Arguments

<i>programType</i>	Program type
<i>parameter</i>	Shader parameter

Return Values

Pointer to a new ParameterValue in the case of changing the reference destination of the parameter value. NULL in the case that there is no change.

Description

This returns the reference destination of a shader parameter. This function is called to all mesh material shader parameters in Node::applyParameterReference(). Though the value specified in the Collada file or the value of the default parameter manager is set to the parameter, for the parameters to which this function returned a value other than NULL, the value of the parameter will be referenced at the time of drawing.

For example, if a valid pointer to ParmeterValue is returned when this function is called to the shader parameter "light," the value of the destination of the returned pointer will be set when all of the values of the "light" parameters are set in GPU at the time of drawing. It is possible to batch-change the values of "light" parameters by changing the value of the pointer destination.

sce::SampleUtil::Graphics::Collada::MatrixChangeListener

Summary

sce::SampleUtil::Graphics::Collada::MatrixChangeListener

Interface to notify the World matrix when drawing a node.

Definition

```
#include <collada_base.h>
class MatrixChangeListener {};
```

Description

If the instance of this interface is set in [ColladaLoader](#), the value of the World matrix of the node will be notified when drawing the node of a scene. To use the value of the World matrix, set the instance of this class.

If [MatrixChangeListener](#) is set in [ColladaLoader](#), the default parameter manager will no longer be notified.

Methods Summary

Methods	Description
setWorldMatrix	Notifies the World matrix of a node.
~MatrixChangeListener	Destructor.

Constructors and Destructors

~MatrixChangeListener

Destructor.

Definition

```
#include <collada_base.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class MatrixChangeListener {
                    virtual inline
~MatrixChangeListener(void);
                };
            }
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Public Instance Methods

setWorldMatrix

Notifies the World matrix of a node.

Definition

```
#include <collada_base.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class MatrixChangeListener {
                    virtual void setWorldMatrix(
                        sce::Vectormath::Simd::Aos::Matrix4<arg> world
                    )=0;
                }
            }
        }
    }
}
```

Arguments

world The world matrix of a node

Return Values

None

Description

This notifies the World matrix of a node.

sce::SampleUtil::Graphics::Collada::DefaultShaderParameterManager

Summary

sce::SampleUtil::Graphics::Collada::DefaultShaderParameterManager

Default shader parameter manager.

Definition

```
#include <collada_base.h>
class DefaultShaderParameterManager :
public sce::SampleUtil::Graphics::Collada::ShaderParameterReference,
public sce::SampleUtil::Graphics::Collada::MatrixChangeListener {};
```

Description

The instance of this class is automatically created in [ColladaLoader](#). The parameters provided by this class are listed. "View" View matrix, "World" World matrix, "WorldInverseTranspose" Inverse transpose matrix of the world matrix, "WorldViewProjection" Projection matrix * View matrix * World matrix, "ViewInverse" Inverse matrix of the view matrix- "BindShapeMatrix" Bind shape matrix, "Light0Position" Position of light 0, "light0Color" Color of light 0.

Methods Summary

Methods	Description
setLight	Sets light 0.
setWorldMatrix	Sets the world matrix.
setViewMatrix	Sets the view matrix.
setProjectionMatrix	Sets the projection matrix.
setShadowTex	Sets the shadow map.
setShadowViewProjection	Sets the projection matrix for the shadow map.

Public Instance Methods

setLight

Sets light 0.

Definition

```
#include <collada_base.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class DefaultShaderParameterManager {
                    virtual void setLight
                        sce::Vectormath::Simd::Aos::Vector3_arg light0Position,
                        sce::Vectormath::Simd::Aos::Vector3_arg light0Color
                            )=0;
                }
            }
        }
    }
}
```

Arguments

<i>light0Position</i>	Light 0 position
<i>light0Color</i>	Light 0 color

Return Values

None

Description

This sets light 0.

SCE CONFIDENTIAL

setWorldMatrix

Sets the world matrix.

Definition

```
#include <collada_base.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class DefaultShaderParameterManager {
                    virtual void setWorldMatrix(
                        sce::Vectormath::Simd::Aos::Matrix4_arg world
                    )=0;
                }
            }
        }
    }
}
```

Arguments

world World matrix

Return Values

None

Description

If [MatrixChangeListener](#) of ColladLoader is not overwritten, the world matrix will be automatically set.

SCE CONFIDENTIAL

setViewMatrix

Sets the view matrix.

Definition

```
#include <collada_base.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class DefaultShaderParameterManager {
                    virtual void setViewMatrix(
                        sce::Vectormath::Simd::Aos::Matrix4_arg view
                    )=0;
                }
            }
        }
    }
}
```

Arguments

view View matrix

Return Values

None

Description

This sets the view matrix.

SCE CONFIDENTIAL

setProjectionMatrix

Sets the projection matrix.

Definition

```
#include <collada_base.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class DefaultShaderParameterManager {
                    virtual void setProjectionMatrix(
                        sce::Vectormath::Simd::Aos::Matrix4_arg projection
                    )=0;
                }
            }
        }
    }
}
```

Arguments

projection Projection matrix

Return Values

None

Description

This sets the projection matrix.

SCE CONFIDENTIAL

setShadowTex

Sets the shadow map.

Definition

```
#include <collada_base.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class DefaultShaderParameterManager {
                    virtual void setShadowTex(
                        Texture *texture
                    )=0;
                }
            }
        }
    }
}
```

Arguments

texture Shadow map

Return Values

None

Description

This sets the shadow map.

SCE CONFIDENTIAL

setShadowViewProjection

Sets the projection matrix for the shadow map.

Definition

```
#include <collada_base.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class DefaultShaderParameterManager {
                    virtual void setShadowViewProjection(
                        sce::Vectormath::Simd::Aos::Matrix4_arg shadowViewProjectionMatrix
                    )=0;
                }
            }
        }
    }
}
```

Arguments

shadowViewProjectionMatrix Projection matrix for a shadow map

Return Values

None

Description

This sets the projection matrix for the shadow map.

sce::SampleUtil::Graphics::Collada::ColladaLoader

Summary

sce::SampleUtil::Graphics::Collada::ColladaLoader

Manages the data shared by multiple Colladas.

Definition

```
#include <collada_base.h>
class ColladaLoader : public sce::SampleUtil::Resource { };
```

Description

This class manages the data shared by multiple Colladas such as the shader for profile_common effect and default parameter manager.

Methods Summary

Methods	Description
load	Reads collada.
getDefaultValueParams	Gets the default shader parameter manager.
getGraphicsLoader	Return graphics loader used at the time of initialization.
getMatrixChangeListener	Returns a world matrix notification object.

Public Instance Methods

load

Reads collada.

Definition

```
#include <collada_base.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class ColladaLoader {
                    virtual int load(
                        sce::SampleUtil::Graphics::Collada::ColladaData **outCollada,
                        const char *daePath,
                        float framePeriod =
                        0.0166666666666666
                    )=0;
                };
            }
        }
    }
}
```

Arguments

<i>outCollada</i>	Pointer to which the generated Collada returns
<i>daePath</i>	Path of a collada file
<i>framePeriod</i>	Sampling cycle of animation

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This reads data from the specified collada file. To delete the generated instance, use [sce::SampleUtil::destroy\(\)](#).

SCE CONFIDENTIAL

getDefaultParams

Gets the default shader parameter manager.

Definition

```
#include <collada_base.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class ColladaLoader {
                    virtual DefaultShaderParameterManager*
*getDefaultParams(void)=0 const;
                };
            }
        }
    }
}
```

Return Values

Default shader parameter manager

Description

This gets the default shader parameter manager.

SCE CONFIDENTIAL

getGraphicsLoader

Return graphics loader used at the time of initialization.

Definition

```
#include <collada_base.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class ColladaLoader {
                    virtual Graphics::GraphicsLoader
*getGraphicsLoader(void)=0 const;
                }
            }
        }
    }
}
```

Return Values

[Graphics](#) loader

Description

This returns the graphics loader used at the time of initialization.

SCE CONFIDENTIAL

getMatrixChangeListener

Returns a world matrix notification object.

Definition

```
#include <collada_base.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class ColladaLoader {
                    virtual MatrixChangeListener
*getMatrixChangeListener(void)=0 const;
                }
            }
        }
    }
}
```

Return Values

World matrix notification object

Description

This returns a world matrix notification object.

sce::SampleUtil::Graphics::Collada::EffectDataParameter

Summary

sce::SampleUtil::Graphics::Collada::EffectDataParameter

[Parameter](#) of an effect.

Definition

```
#include <collada_effect.h>
class EffectDataParameter { };
```

Description

This is the parameter of an effect

Methods Summary

Methods	Description
getName	Gets a parameter name.
getType	Gets a parameter type.
getFloatArray	Gets a parameter value as the array of float.
getFloatArraySize	Gets the size of the float array of a parameter value.
getImageId	Image ID of a parameter value.
EffectDataParameter	Constructor.
operator=	Copy operator.
~EffectDataParameter	Destructor.

Enumerated Types

Type

Datatype.

Definition

```
#include <collada_effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class EffectDataParameter {
                    enum Type {
                        kTypeFloatArray,
                        kTypeImage
                    };
                }
            }
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kTypeFloatArray	N/A	Array of float.
kTypeImage	N/A	Image .

Description

This is a datatype.

Constructors and Destructors

EffectDataParameter

Constructor.

Definition

```
#include <collada_effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class EffectDataParameter {
                    inline EffectDataParameter(void);
                }
            }
        }
    }
}
```

Return Values

None

Description

This initialized as a blank parameter.

SCE CONFIDENTIAL

EffectDataParameter

Constructor.

Definition

```
#include <collada_effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class EffectDataParameter {
                    inline EffectDataParameter(
                        std::string name,
                        const float *value,
                        uint32_t size
                    );
                };
            }
        }
    }
}
```

Arguments

<i>name</i>	Parameter name
<i>value</i>	float data
<i>size</i>	Size of float data

Return Values

None

Description

This initializes as data with a float array. The float data is copied in.

SCE CONFIDENTIAL

EffectDataParameter

Constructor.

Definition

```
#include <collada_effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class EffectDataParameter {
                    inline EffectDataParameter(
                        std::string name,
                        std::string imageId
                    );
                };
            }
        }
    }
}
```

Arguments

name [Parameter](#) name
imageId [Image](#) ID

Return Values

None

Description

This initializes as a parameter with an image.

SCE CONFIDENTIAL

EffectDataParameter

Constructor.

Definition

```
#include <collada_effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class EffectDataParameter {
                    inline EffectDataParameter(
                        const EffectDataParameter &param
                    );
                }
            }
        }
    }
}
```

Arguments

param The data of a copy source

Return Values

None

Description

This copies the value of an argument.

SCE CONFIDENTIAL

~EffectDataParameter

Destructor.

Definition

```
#include <collada_effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class EffectDataParameter {
                    inline ~EffectDataParameter(void);
                }
            }
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Operator Methods

operator=

Copy operator.

Definition

```
#include <collada_effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class EffectDataParameter {
                    inline EffectDataParameter &operator=(const EffectDataParameter &param);
                };
            }
        }
    }
}
```

Arguments

param The data of a copy source

Return Values

The reference of this instance

Description

This copies the value of an argument.

Public Instance Methods

getName

Gets a parameter name.

Definition

```
#include <collada_effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class EffectDataParameter {
                    inline std::string getName(void) const;
                }
            }
        }
    }
}
```

Return Values

[Parameter](#) name

Description

This gets a parameter name.

SCE CONFIDENTIAL

getType

Gets a parameter type.

Definition

```
#include <collada_effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class EffectDataParameter {
                    inline Type getType(void) const;
                }
            }
        }
    }
}
```

Return Values

[Parameter](#) type

Description

This gets a parameter type.

SCE CONFIDENTIAL

getFloatArray

Gets a parameter value as the array of float.

Definition

```
#include <collada_effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class EffectDataParameter {
                    inline const float *getFloatArray(void)
const;
```

Return Values

[Parameter](#) value. NULL in the case that the type of the parameter is not kTypeFloatArray.

Description

This gets a parameter value as the array of float.

SCE CONFIDENTIAL

getFloatArraySize

Gets the size of the float array of a parameter value.

Definition

```
#include <collada_effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class EffectDataParameter {
                    inline uint32_t getFloatArraySize(void)
const;
```

Return Values

Size of the float array of a parameter value. 0 in the case that the type of the parameter is not TYPE_FLOAT_ARRAY

Description

This gets the size of the float array of a parameter value.

getImageId

[Image](#) ID of a parameter value.

Definition

```
#include <collada_effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class EffectDataParameter {
                    inline std::string getImageId(void) const;
                }
            }
        }
    }
}
```

Return Values

[Image](#) ID of a parameter value. Blank character string in the case that the type of the parameter is not TYPE_IMAGE

Description

This gets the image ID of a parameter value.

sce::SampleUtil::Graphics::Collada::EffectDataShader

Summary

sce::SampleUtil::Graphics::Collada::EffectDataShader

The shader information of an effect.

Definition

```
#include <collada_effect.h>
class EffectDataShader {};
```

Description

This is the shader information of an effect.

Methods Summary

Methods	Description
getPath	Gets the path of a shader.
getEntry	Gets the entry function name of a shader.
getNumParameters	Gets the array of a parameter.
addParameter	Adds a parameter.

Public Instance Methods

getPath

Gets the path of a shader.

Definition

```
#include <collada_effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class EffectDataShader {
                    virtual std::string getPath(void)=0 const;
                }
            }
        }
    }
}
```

Return Values

Path of a shader

Description

This gets the path of a shader.

SCE CONFIDENTIAL

getEntry

Gets the entry function name of a shader.

Definition

```
#include <collada_effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class EffectDataShader {
                    virtual std::string getEntry(void)=0
                const;
            }
        }
    }
}
```

Return Values

The entry function name of a shader

Description

This gets the entry function name of a shader.

SCE CONFIDENTIAL

getNumParameters

Gets the array of a parameter.

Definition

```
#include <collada_effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class EffectDataShader {
                    virtual uint32_t getNumParameters(void)=0
                const;
            }
        }
    }
}
```

Return Values

Array of a parameter

Description

This gets the array of a parameter.

SCE CONFIDENTIAL

addParameter

Adds a parameter.

Definition

```
#include <collada_effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class EffectDataShader {
                    virtual void addParameter(
                        const EffectDataParameter &param
                    )=0;
                }
            }
        }
    }
}
```

Arguments

param [Parameter](#)

Return Values

None

Description

This adds a parameter. The parameter is copied in internally.

sce::SampleUtil::Graphics::Collada::EffectData

Summary

sce::SampleUtil::Graphics::Collada::EffectData

[Effect](#) data.

Definition

```
#include <collada_effect.h>
class EffectData {};
```

Description

This is data to create an effect.

Methods Summary

Methods	Description
getVertexShader	Gets the vertex shader.
getFragmentShader	Gets the fragment shader.
getVertexShader	Gets the vertex shader.
getFragmentShader	Gets the fragment shader.

Public Instance Methods

getVertexShader

Gets the vertex shader.

Definition

```
#include <collada_effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class EffectData {
                    virtual EffectDataShader
*getVertexShader(void)=0;
                }
            }
        }
    }
}
```

Return Values

Vertex shader

Description

This gets the vertex shader.

SCE CONFIDENTIAL

getFragmentShader

Gets the fragment shader.

Definition

```
#include <collada_effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class EffectData {
                    virtual EffectDataShader*
*getFragmentShader(void)=0;
                }
            }
        }
    }
}
```

Return Values

Fragment shader

Description

This gets the fragment shader.

SCE CONFIDENTIAL

getVertexShader

Gets the vertex shader.

Definition

```
#include <collada_effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class EffectData {
                    virtual const EffectDataShader*
*getVertexShader(void)=0 const;
                }
            }
        }
    }
}
```

Return Values

Vertex shader

Description

This gets the vertex shader.

SCE CONFIDENTIAL

getFragmentShader

Gets the fragment shader.

Definition

```
#include <collada_effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class EffectData {
                    virtual const EffectDataShader*
*getFragmentShader(void)=0 const;
                }
            }
        }
    }
}
```

Return Values

Fragment shader

Description

This gets the fragment shader.

sce::SampleUtil::Graphics::Collada::Material

000004892117

Summary

sce::SampleUtil::Graphics::Collada::Material

[Material](#) class.

Definition

```
#include <collada_effect.h>
class Material {};
```

Description

This is a material class

Methods Summary

Methods	Description
getId	Gets the ID of a material class.
getEffect	Gets an effect.
getEffectData	Gets the information of an effect.

Public Instance Methods

getId

Gets the ID of a material class.

Definition

```
#include <collada_effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Material {
                    virtual std::string getId(void)=0 const;
                }
            }
        }
    }
}
```

Return Values

ID of a material

Description

This gets the number of materials.

SCE CONFIDENTIAL

getEffect

Gets an effect.

Definition

```
#include <collada_effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Material {
                    virtual
                const Graphics::Effect::EffectData *getEffect(void)=0 const;
            }
        }
    }
}
```

Return Values

[Effect](#)

Description

This gets an effect.

SCE CONFIDENTIAL

getEffectData

Gets the information of an effect.

Definition

```
#include <collada_effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Material {
                    virtual const EffectData
*getEffectData(void)=0 const;
                }
            }
        }
    }
}
```

Return Values

[Effect](#) data

Description

This returns the data which was the generation source of an effect.

**sce::SampleUtil::Graphics::Collada::Me
shPart**

Summary

sce::SampleUtil::Graphics::Collada::MeshPart

[Mesh](#) part class.

Definition

```
#include <collada_geometry.h>
class MeshPart {};
```

Description

This is the mesh part class.

Methods Summary

Methods	Description
getVertexAttribute	Gets the information of an attribute parameter.
getVertexAttribute	Gets the information of an attribute parameter.
getNumVertexAttributes	Gets the number of attribute parameters.
getStride	Gets the stride of vertex data.
getVertexBuffer	Gets a vertex buffer.
getIndexBuffer	Gets an index buffer.
getNumIndices	Gets the number of indices.
getNumVertices	Gets the number of vertexes.
getMaterial	Returns the reference name of a material.
hasSkin	Confirms whether a skin is included.

Public Instance Methods

getVertexAttribute

Gets the information of an attribute parameter.

Definition

```
#include <collada_geometry.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class MeshPart {
                    Graphics::VertexAttribute
                    getVertexAttribute(ParameterSemantic
semantics,
semanticIndex = 0
) const;
                }
            }
        }
    }
}
```

Arguments

<i>semantic</i>	Semantic of an attribute parameter
<i>semanticIndex</i>	Semantic index of attribute parameters

Return Values

The information of an attribute parameter. If not found, a [VertexAttribute](#) in which everything is set to 0 will be returned.

Description

This gets the information of the vertex buffer specified by a semantic and semantic index. streamIndex is set to 0.

SCE CONFIDENTIAL

getVertexAttribute

Gets the information of an attribute parameter.

Definition

```
#include <collada_geometry.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class MeshPart {
                    Graphics::VertexAttribute
                    getVertexAttribute(
                        uint32_t index
                    ) const;
                }
            }
        }
    }
}
```

Arguments

index Index

Return Values

The information of an attribute parameter. If index is equal to or larger than the number of attribute parameters, [VertexAttribute](#) in which everything is set to 0 will be returned.

Description

This gets the information of a vertex buffer. streamIndex is set to 0.

SCE CONFIDENTIAL

getNumVertexAttributes

Gets the number of attribute parameters.

Definition

```
#include <collada_geometry.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class MeshPart {
                    uint32_t getNumVertexAttributes(void)
                const;
            }
        }
    }
}
```

Return Values

Number of attribute parameters.

Description

This gets the number of attribute parameters.

getStride

Gets the stride of vertex data.

Definition

```
#include <collada_geometry.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class MeshPart {
                    uint32_t getStride(void) const;
                }
            }
        }
    }
}
```

Return Values

Stride of vertex data

Description

This gets the stride of vertex data.

SCE CONFIDENTIAL

getVertexBuffer

Gets a vertex buffer.

Definition

```
#include <collada_geometry.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class MeshPart {
                    Graphics::VertexBuffer
                    *getVertexBuffer(void) const;
                }
            }
        }
    }
}
```

Return Values

Vertex buffer

Description

This gets a vertex buffer.

SCE CONFIDENTIAL

getIndexBuffer

Gets an index buffer.

Definition

```
#include <collada_geometry.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class MeshPart {
                    Graphics::IndexBuffer
                    *getIndexBuffer(void) const;
                }
            }
        }
    }
}
```

Return Values

Index buffer.

Description

This gets an index buffer.

SCE CONFIDENTIAL

getNumIndices

Gets the number of indices.

Definition

```
#include <collada_geometry.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class MeshPart {
                    uint32_t getNumIndices(void) const;
                }
            }
        }
    }
}
```

Return Values

Number of indices

Description

This gets the number of indices.

SCE CONFIDENTIAL

getNumVertices

Gets the number of vertexes.

Definition

```
#include <collada_geometry.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class MeshPart {
                    uint32_t getNumVertices(void) const;
                }
            }
        }
    }
}
```

Return Values

Number of vertexes

Description

This gets the number of vertexes.

getMaterial

Returns the reference name of a material.

Definition

```
#include <collada_geometry.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class MeshPart {
                    std::string getMaterial(void) const;
                }
            }
        }
    }
}
```

Return Values

The reference name of a material

Description

This returns the reference name of a material to be used for drawing this mesh part. The association between the reference name of a material and the material is specified by [InstanceMesh](#).

SCE CONFIDENTIAL

hasSkin

Confirms whether a skin is included.

Definition

```
#include <collada_geometry.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class MeshPart {
                    bool hasSkin(void) const;
                }
            }
        }
    }
}
```

Return Values

true will be returned if a skin is included, and false if not.

Description

This confirms whether a skin is included in this mesh part.

**sce::SampleUtil::Graphics::Collada::Me
sh**

Summary

sce::SampleUtil::Graphics::Collada::Mesh

[Mesh](#) class.

Definition

```
#include <collada_geometry.h>
class Mesh {};
```

Description

This is the mesh class.

Methods Summary

Methods	Description
hasSkin	Confirms whether a skin is included in a mesh.
getMeshPart	Gets a mesh part.
getNumMeshParts	Gets the number of mesh parts.
getControllerId	Get the controller ID.
getGeometryId	Gets a geometry ID.
getJointNameType	Gets the type of the name of a joint name array.
getNumJoints	Gets a joint count.
getJointName	Gets a joint name.
getInvBindMatrices	Gets an inverse bind matrix.
getBindShapeMatrix	Gets a bind shape matrix.

Enumerated Types

JointNameType

Type of the name of a joint name array.

Definition

```
#include <collada_geometry.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Mesh {
                    enum JointNameType {
                        kJointNameTypeId,
                        kJointNameTypeSid
                    };
                }
            }
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kJointNameTypeId	N/A	getJointNameArray() is the array of the ID of a joint node
kJointNameTypeSid	N/A	getJointNameArray() is the array of the SID of a joint node

Description

This is the type to indicate the type of the value of the joint name array which can be acquired by getJointNameArray().

Public Instance Methods

hasSkin

Confirms whether a skin is included in a mesh.

Definition

```
#include <collada_geometry.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Mesh {
                    virtual bool hasSkin(void)=0 const;
                }
            }
        }
    }
}
```

Return Values

true will be returned if a skin is included, and false if not.

Description

This confirms whether a skin is included in a mesh.

getMeshPart

Gets a mesh part.

Definition

```
#include <collada_geometry.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Mesh {
                    virtual const MeshPart *getMeshPart(
                        uint32_t index
                    )=0 const;
                }
            }
        }
    }
}
```

Arguments

index Index of mesh parts

Return Values

[Mesh](#) part. If index is equal to or larger than the number of mesh parts, NULL will be returned.

Description

This gets a mesh part.

SCE CONFIDENTIAL

getNumMeshParts

Gets the number of mesh parts.

Definition

```
#include <collada_geometry.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Mesh {
                    virtual uint32_t getNumMeshParts(void)=0
                const;
            }
        }
    }
}
```

Return Values

Number of mesh parts

Description

This gets the number of mesh parts.

SCE CONFIDENTIAL

getControllerId

Get the controller ID.

Definition

```
#include <collada_geometry.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Mesh {
                    virtual std::string
getControllerId(void)=0 const;
                }
            }
        }
    }
}
```

Return Values

Controller ID

Description

This function obtains the controller ID.

SCE CONFIDENTIAL

getGeometryId

Gets a geometry ID.

Definition

```
#include <collada_geometry.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Mesh {
                    virtual std::string getGeometryId(void)=0
                const;
            }
        }
    }
}
```

Return Values

Geometry ID

Description

This gets a geometry ID.

SCE CONFIDENTIAL

getJointNameType

Gets the type of the name of a joint name array.

Definition

```
#include <collada_geometry.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Mesh {
                    virtual JointNameType
getJointNameType(void)=0 const;
                }
            }
        }
    }
}
```

Return Values

Type of the name of a joint name array

Description

This gets the type of the name of a joint name array.

SCE CONFIDENTIAL

getNumJoints

Gets a joint count.

Definition

```
#include <collada_geometry.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Mesh {
                    virtual uint32_t getNumJoints(void)=0
                const;
            }
        }
    }
}
```

Return Values

Joint count

Description

This gets a joint count.

SCE CONFIDENTIAL

getJointName

Gets a joint name.

Definition

```
#include <collada_geometry.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Mesh {
                    virtual std::string getJointName(
                        uint32_t jointIndex
                    )=0 const;
                }
            }
        }
    }
}
```

Arguments

jointIndex Joint index

Return Values

Joint name

Description

This gets a joint name.

SCE CONFIDENTIAL

getInvBindMatrices

Gets an inverse bind matrix.

Definition

```
#include <collada_geometry.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Mesh {
                    virtual const
sce::Vectormath::Simd::Aos::Matrix4 *getInvBindMatrices(void)=0 const;
                }
            }
        }
    }
}
```

Return Values

Array of an inverse bind matrix

Description

This gets an inverse bind matrix.

SCE CONFIDENTIAL

getBindShapeMatrix

Gets a bind shape matrix.

Definition

```
#include <collada_geometry.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Mesh {
                    virtual
sce::Vectormath::Simd::Aos::Matrix4 getBindShapeMatrix(void)=0 const;
                }
            }
        }
    }
}
```

Return Values

Bind shape matrix

Description

This gets a bind shape matrix.

sce::SampleUtil::Graphics::Collada::Image

000004892117

Summary

sce::SampleUtil::Graphics::Collada::Image

[Image](#) class.

Definition

```
#include <collada_image.h>
class Image {};
```

Description

This is the image class.

Methods Summary

Methods	Description
getTexture	Gets a texture.
getTexture	Gets a texture.
getId	Gets an ID.
getName	Gets a name.
getAbsolutePath	Absolute path of an image.
getPath	Path of an image.

Public Instance Methods

getTexture

Gets a texture.

Definition

```
#include <collada_image.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Image {
                    sce::SampleUtil::Graphics::Texture
                    *getTexture(void);
                }
            }
        }
    }
}
```

Return Values

[Texture](#)

Description

This returns the texture of this image.

SCE CONFIDENTIAL

getTexture

Gets a texture.

Definition

```
#include <collada_image.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Image {
                    const sce::SampleUtil::Graphics::Texture
*getTexture(void) const;
                }
            }
        }
    }
}
```

Return Values

[Texture](#)

Description

This returns the texture of this image.

SCE CONFIDENTIAL

getId

Gets an ID.

Definition

```
#include <collada_image.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Image {
                    const std::string getId(void) const;
                }
            }
        }
    }
}
```

Return Values

ID

Description

This returns the ID of this image.

SCE CONFIDENTIAL

getName

Gets a name.

Definition

```
#include <collada_image.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Image {
                    const std::string getName(void) const;
                }
            }
        }
    }
}
```

Return Values

Name

Description

This returns the name of this image.

SCE CONFIDENTIAL

getAbsolutePath

Absolute path of an image.

Definition

```
#include <collada_image.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Image {
                    const std::string getAbsolutePath(void)
                };
            }
        }
    }
}
```

Return Values

Absolute path

Description

This returns the absolute path of this image.

SCE CONFIDENTIAL

getPath

Path of an image.

Definition

```
#include <collada_image.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Image {
                    const std::string getPath(void) const;
                }
            }
        }
    }
}
```

Return Values

Path

Description

This returns the path of this image. The absolute path or the relative path from a collada file is returned.

sce::SampleUtil::Graphics::Collada::InstanceMeshPart

Summary

sce::SampleUtil::Graphics::Collada::InstanceMeshPart

Instance of a mesh part.

Definition

```
#include <collada_scene.h>
class InstanceMeshPart {};
```

Description

This is the instance of a mesh part.

Methods Summary

Methods	Description
draw	Draws a mesh part.
applyShaderParameterManager	Changes the reference destination of a shader parameter.
addEffect	Adds an effect.

Public Instance Methods

draw

Draws a mesh part.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class InstanceMeshPart {
                    virtual void draw(
                        Graphics::GraphicsContext
                        Graphics::UniformBuffer
                        int renderingStage
                    )=0;
                }
            }
        }
    }
}
```

Arguments

graphicsContext
skinMatrices
renderingStage

[Graphics](#) context

Skin matrices. It is required only if a mesh has a skin.

Rendering stage. This draws only effect whose rendering stage matches.

Return Values

None

Description

This draws a mesh part.

applyShaderParameterManager

Changes the reference destination of a shader parameter.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class InstanceMeshPart {
                    virtual void applyShaderParameterManager(
                        ShaderParameterReference
                        *nodeParameterManager
                    )=0;
                }
            }
        }
    }
}
```

Arguments

nodeParameterManager The reference destination of a shader parameter

Return Values

None

Description

This changes the reference destination of a parameter used for shader drawing in a scene. `getParamValue()` of reference is called to the shader parameters of all effects included in this mesh part, and the reference destination of a parameter to which a value other than NULL was returned is changed.

SCE CONFIDENTIAL

addEffect

Adds an effect.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class InstanceMeshPart {
                    virtual int addEffect(
                        const Graphics::Effect::EffectData*
                        *effect,
                        int renderingStage
                    )=0;
                }
            }
        }
    }
}
```

Arguments

<i>effect</i>	The effect to be added
<i>renderingStage</i>	Rendering stage to draw

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This adds an effect. This is used for drawing only when renderingStage specified by draw matches.

sce::SampleUtil::Graphics::Collada::InstanceMesh

Summary

sce::SampleUtil::Graphics::Collada::InstanceMesh

Instance of a mesh.

Definition

```
#include <collada_scene.h>
class InstanceMesh {};
```

Description

This is the instance of a mesh.

Methods Summary

Methods	Description
updateSkinMatrices	Updates the skin matrix of a mesh.
applyShaderParameterManager	Changes the reference destination of a shader parameter.
draw	Drawing.
getNumMeshParts	Gets the number of mesh parts.
getMeshPart	Gets a mesh part.
getMesh	Gets the mesh which was the source.
addEffect	Adds an effect.

Public Instance Methods

updateSkinMatrices

Updates the skin matrix of a mesh.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class InstanceMesh {
                    virtual void updateSkinMatrices(void)=0;
                }
            }
        }
    }
}
```

Return Values

None

Description

This updates the skin matrix of a mesh. The world matrix of the node to be referenced needs to be updated before this function is called.

SCE CONFIDENTIAL

applyShaderParameterManager

Changes the reference destination of a shader parameter.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class InstanceMesh {
                    virtual void applyShaderParameterManager(
                        ShaderParameterReference
                        *nodeParameterManager
                    )=0;
                }
            }
        }
    }
}
```

Arguments

nodeParameterManager The reference destination of a shader parameter

Return Values

None

Description

This changes the reference destination of a parameter used for shader drawing in a scene. `getParamValue()` of reference is called to the shader parameters of all mesh parts included in this mesh, and the reference destination of a parameter to which a value other than NULL was returned is changed.

SCE CONFIDENTIAL

draw

Drawing.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class InstanceMesh {
                    virtual void draw(
                        sce::Vectormath::Simd::Aos::Matrix4_arg localToWorld,
                        Graphics::GraphicsContext
                        *graphicsContext,
                        int renderingStage
                    )=0;
                }
            }
        }
    }
}
```

Arguments

<i>localToWorld</i>	The transformation matrix from the local coordinate of this mesh to the world coordinate
<i>graphicsContext</i>	Graphics context
<i>renderingStage</i>	Rendering stage. This draws only effect whose rendering stage matches.

Return Values

None

Description

This draws all mesh parts included in this mesh.

SCE CONFIDENTIAL

getNumMeshParts

Gets the number of mesh parts.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class InstanceMesh {
                    virtual uint32_t getNumMeshParts(void)=0
                const;
            }
        }
    }
}
```

Return Values

Number of mesh parts

Description

This gets the number of mesh parts.

getMeshPart

Gets a mesh part.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class InstanceMesh {
                    virtual InstanceMeshPart *getMeshPart(
                        uint32_t index
                    )=0;
                };
            }
        }
    }
}
```

Arguments

index Index of mesh parts

Return Values

[Mesh](#) part. NULL in the case that index is equal to or larger than the number of mesh parts.

Description

This gets a mesh part.

SCE CONFIDENTIAL

getMesh

Gets the mesh which was the source.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class InstanceMesh {
                    virtual const Mesh *getMesh(void)=0 const;
                }
            }
        }
    }
}
```

Return Values

[Mesh](#)

Description

This returns the mesh which was the generation source of this instance mesh.

SCE CONFIDENTIAL

addEffect

Adds an effect.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class InstanceMesh {
                    virtual int addEffect(
                        const Graphics::Effect::EffectData*
                        *effect,
                        int renderingStage
                    )=0;
                }
            }
        }
    }
}
```

Arguments

<i>effect</i>	The effect to be added
<i>renderingStage</i>	Rendering stage to draw

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This adds an effect to all mesh parts included in this mesh. This is used for drawing only when renderingStage specified by draw matches.

**sce::SampleUtil::Graphics::Collada::No
de**

000004892117

Summary

sce::SampleUtil::Graphics::Collada::Node

Node of a scene.

Definition

```
#include <collada_scene.h>
class Node {};
```

Description

This is the node of a scene.

Methods Summary

Methods	Description
getId	Returns the ID of a node.
getSid	Returns the SID of a node.
findNodeBySid	Searches the node with the SID specified from the subtree lower than this node.
findNodeById	Searches the node with the ID specified from the subtree lower than this node.
findNodeBySid	Searches the node with the SID specified from the subtree lower than this node.
findNodeById	Searches the node with the ID specified from the subtree lower than this node.
getTransformMatrix	Returns the transformation matrix of this node.
setTransformMatrix	Sets the transformation matrix of a node.
getWorldMatrix	Returns the world matrix of a node with the root node as the starting point.
getWorldMatrixPointer	Returns the pointer to the world matrix of a node with the root node as the starting point.
updateWorldMatrix	Updates the world matrix of a node.
updateSkinMatrices	Updates the skin matrices of the meshes under this node.
draw	Draws the subtrees under this node.
applyShaderParameterManager	Changes the reference destination of a shader parameter.
getInstanceMesh	Gets an instance mesh.
getNumInstanceMeshes	Gets the number of instance meshes.
getNumChildNodes	Gets the number of child nodes.
getChildNode	Returns a child node.
getChildNode	Returns a child node.
addEffect	Adds an effect.

Public Instance Methods

getId

Returns the ID of a node.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Node {
                    virtual std::string getId(void)=0 const;
                }
            }
        }
    }
}
```

Return Values

ID of a node

Description

This returns the ID of a node.

SCE CONFIDENTIAL

getSid

Returns the SID of a node.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Node {
                    virtual std::string getSid(void)=0 const;
                }
            }
        }
    }
}
```

Return Values

SID of a node

Description

This returns the SID of a node.

SCE CONFIDENTIAL

findNodeBySid

Searches the node with the SID specified from the subtree lower than this node.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Node {
                    virtual Node *findNodeBySid(
                        std::string sid
                    )=0;
                }
            }
        }
    }
}
```

Arguments

sid SID of a node

Return Values

[Node](#) with the specified SID. NULL if not found.

Description

This searches the node with the SID specified from the subtree lower than this node.

SCE CONFIDENTIAL

findNodeById

Searches the node with the ID specified from the subtree lower than this node.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Node {
                    virtual Node *findNodeById(
                        std::string id
                    )=0;
                }
            }
        }
    }
}
```

Arguments

id ID of a node

Return Values

[Node](#) with the specified ID. NULL if not found.

Description

This searches the node with the ID specified from the subtree lower than this node.

findNodeBySid

Searches the node with the SID specified from the subtree lower than this node.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Node {
                    virtual const Node *findNodeBySid(
                        std::string sid
                    )=0 const;
                }
            }
        }
    }
}
```

Arguments

sid SID of a node

Return Values

[Node](#) with the specified SID. NULL if not found.

Description

This searches the node with the SID specified from the subtree lower than this node.

SCE CONFIDENTIAL

findNodeById

Searches the node with the ID specified from the subtree lower than this node.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Node {
                    virtual const Node *findNodeById(
                        std::string id
                    )=0 const;
                }
            }
        }
    }
}
```

Arguments

id ID of a node

Return Values

[Node](#) with the specified ID. NULL if not found.

Description

This searches the node with the ID specified from the subtree lower than this node.

SCE CONFIDENTIAL

getTransformMatrix

Returns the transformation matrix of this node.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Node {
                    virtual
sce::Vectormath::Simd::Aos::Matrix4 getTransformMatrix(void)=0 const;
                }
            }
        }
    }
}
```

Return Values

Transformation matrix of a node

Description

This returns the transformation matrix of this node.

SCE CONFIDENTIAL

setTransformMatrix

Sets the transformation matrix of a node.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Node {
                    virtual void setTransformMatrix(
                        sce::Vectormath::Simd::Aos::Matrix4_arg matrix
                    )=0;
                }
            }
        }
    }
}
```

Arguments

matrix Transformation matrix of a node

Return Values

None

Description

This sets the transformation matrix of a node.

SCE CONFIDENTIAL

getWorldMatrix

Returns the world matrix of a node with the root node as the starting point.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Node {
                    virtual
sce::Vectormath::Simd::Aos::Matrix4 getWorldMatrix(void)=0 const;
                }
            }
        }
    }
}
```

Return Values

The world matrix of the world matrix node of a node

Description

This returns the world matrix of a node with the root node as the starting point.

SCE CONFIDENTIAL

getWorldMatrixPointer

Returns the pointer to the world matrix of a node with the root node as the starting point.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Node {
                    virtual const
sce::Vectormath::Simd::Aos::Matrix4 *getWorldMatrixPointer(void)=0 const;
                }
            }
        }
    }
}
```

Return Values

Pointer to the world matrix of the world matrix node of a node

Description

This returns the pointer to the world matrix of a node with the root node as the starting point.

SCE CONFIDENTIAL

updateWorldMatrix

Updates the world matrix of a node.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Node {
                    virtual void updateWorldMatrix(void)=0;
                }
            }
        }
    }
}
```

Return Values

None

Description

This updates the world matrix of a node. This calculates the world matrix of this node from the world matrix of the parent node and the transformation matrix of this node, and sets it. Also this updates the world matrix of a child node recursively.

SCE CONFIDENTIAL

updateSkinMatrices

Updates the skin matrices of the meshes under this node.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Node {
                    virtual void updateSkinMatrices(void)=0;
                }
            }
        }
    }
}
```

Return Values

None

Description

This updates the skin matrices of the meshes under this node. This updates the skin matrices of the meshes included in the subtrees under this node. The world matrix needs to be updated before this function is called.

SCE CONFIDENTIAL

draw

Draws the subtrees under this node.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Node {
                    virtual void draw(
                        GraphicsContext *graphicsContext,
                        sce::Vectormath::Simd::Aos::Matrix4_arg matrix,
                        int renderingStage
                    )=0 const;
                }
            }
        }
    }
}
```

Arguments

graphicsContext
matrix

renderingStage

[Graphics](#) context

Transformation matrix. The product of matrix and the world matrix of a node is given to a shader as the world matrix.

Rendering stage. This draws only effect of the specified rendering stage.

Return Values

None

Description

This draws the subtrees under this node.

applyShaderParameterManager

Changes the reference destination of a shader parameter.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Node {
                    virtual void applyShaderParameterManager(
                        ShaderParameterReference
                        *nodeParameterManager
                    )=0;
                }
            }
        }
    }
}
```

Arguments

nodeParameterManager The reference destination of a shader parameter

Return Values

None

Description

This changes the reference destination of a parameter used for shader drawing in a scene. `getParamValue()` of reference is called to all shader parameters in the subtrees under this node, and the reference destination of a parameter to which a value other than NULL was returned is changed.

getInstanceMesh

Gets an instance mesh.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Node {
                    virtual InstanceMesh *getInstanceMesh(
                        uint32_t index
                    )=0;
                }
            }
        }
    }
}
```

Arguments

index Index of instance meshes.

Return Values

Instance mesh. NULL in the case that index larger than the number of instance meshes is specified.

Description

This returns the instance mesh which this node has.

SCE CONFIDENTIAL

getNumInstanceMeshes

Gets the number of instance meshes.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Node {
                    virtual uint32_t
getNumInstanceMeshes(void)=0 const;
                }
            }
        }
    }
}
```

Return Values

Number of instance meshes

Description

This returns the number of instance meshes which this node has.

SCE CONFIDENTIAL

getNumChildNodes

Gets the number of child nodes.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Node {
                    virtual uint32_t getNumChildNodes(void)=0
                const;
            }
        }
    }
}
```

Return Values

Number of child nodes

Description

This returns the number of child nodes.

SCE CONFIDENTIAL

getChildNode

Returns a child node.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Node {
                    virtual Node *getChildNode(
                        uint32_t index
                    )=0;
                }
            }
        }
    }
}
```

Arguments

index Index of child nodes.

Return Values

Child node. NULL in the case that index larger than the number of child nodes is specified.

Description

This returns a child node.

SCE CONFIDENTIAL

getTreeNode

Returns a child node.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Node {
                    virtual const Node *getTreeNode(
                        uint32_t index
                    )=0 const;
                }
            }
        }
    }
}
```

Arguments

index Index of child nodes.

Return Values

Child node. NULL in the case that index larger than the number of child nodes is specified.

Description

This returns a child node.

SCE CONFIDENTIAL

addEffect

Adds an effect.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Node {
                    virtual int addEffect(
                        const Graphics::Effect::EffectData*
                        *effect,
                        int renderingStage
                    )=0;
                }
            }
        }
    }
}
```

Arguments

<i>effect</i>	The effect to be added
<i>renderingStage</i>	Rendering stage to draw

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This adds the specified effect to all meshes of the subtrees under this node. This is used for drawing only when renderingStage specified by draw matches.

sce::SampleUtil::Graphics::Collada::Animation

Summary

sce::SampleUtil::Graphics::Collada::Animation

[Animation.](#)

Definition

```
#include <collada_scene.h>
class Animation {};
```

Description

This is the class to retain the node animation data of a scene.

Methods Summary

Methods	Description
getStartTime	Returns the starting time of animation.
getEndTime	Returns the finish time of animation.
getNumFrames	Returns the number of frames.
getNumAnimationNodes	Returns the number of animation nodes.
getAnimationNode	Returns the animation node of the specified index.

Public Instance Methods

getStartTime

Returns the starting time of animation.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Animation {
                    float getStartTime(void) const;
                }
            }
        }
    }
}
```

Return Values

The starting time of animation

Description

This returns the starting time of animation.

SCE CONFIDENTIAL

getEndTime

Returns the finish time of animation.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Animation {
                    float getEndTime(void) const;
                }
            }
        }
    }
}
```

Return Values

The finish time of animation

Description

This returns the finish time of animation.

SCE CONFIDENTIAL

getNumFrames

Returns the number of frames.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Animation {
                    uint32_t getNumFrames(void) const;
                }
            }
        }
    }
}
```

Return Values

Number of frames

Description

This returns the number of frames.

SCE CONFIDENTIAL

getNumAnimationNodes

Returns the number of animation nodes.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Animation {
                    inline uint32_t
getNumAnimationNodes(void) const;
                }
            }
        }
    }
}
```

Return Values

Number of animation nodes

Description

This returns the number of animation nodes.

SCE CONFIDENTIAL

getAnimationNode

Returns the animation node of the specified index.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class Animation {
                    inline const NodeAnimation*
*getAnimationNode(
                        uint32_t index
                    ) const;
                }
            }
        }
    }
}
```

Arguments

index Index

Return Values

[Animation](#) node

Description

This returns the animation node of the specified index.

sce::SampleUtil::Graphics::Collada::VisualScene

Summary

sce::SampleUtil::Graphics::Collada::VisualScene

Scene class.

Definition

```
#include <collada_scene.h>
class VisualScene {};
```

Description

This is the scene class.

Methods Summary

Methods	Description
<u>getId</u>	Returns the ID of a scene.
<u>getName</u>	Returns the name of a scene.
<u>findNodeById</u>	Returns the node with the specified ID.
<u>findNodeBySid</u>	Returns the node with the specified SID.
<u>findNodeById</u>	Returns the node with the specified ID.
<u>findNodeBySid</u>	Returns the node with the specified SID.
<u>get rootNode</u>	Returns the root node.
<u>get rootNode</u>	Returns the root node.

Public Instance Methods

getId

Returns the ID of a scene.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class VisualScene {
                    virtual std::string getId(void)=0 const;
                }
            }
        }
    }
}
```

Return Values

ID

Description

This returns the ID of a scene.

SCE CONFIDENTIAL

getName

Returns the name of a scene.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class VisualScene {
                    virtual std::string getName(void)=0 const;
                }
            }
        }
    }
}
```

Return Values

Name

Description

This returns the name of a scene.

SCE CONFIDENTIAL

findNodeById

Returns the node with the specified ID.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class VisualScene {
                    virtual Node *findNodeById(
                        std::string id
                    )=0;
                }
            }
        }
    }
}
```

Arguments

id ID

Return Values

[Node](#). NULL if not found.

Description

This returns the node with the specified ID.

SCE CONFIDENTIAL

findNodeBySid

Returns the node with the specified SID.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class VisualScene {
                    virtual Node *findNodeBySid(
                        std::string sid
                    )=0;
                }
            }
        }
    }
}
```

Arguments

sid SID

Return Values

[Node](#). NULL if not found.

Description

This returns the node with the specified SID.

SCE CONFIDENTIAL

findNodeById

Returns the node with the specified ID.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class VisualScene {
                    virtual const Node *findNodeById(
                        std::string id
                    )=0 const;
                }
            }
        }
    }
}
```

Arguments

id ID

Return Values

[Node](#). NULL if not found.

Description

This returns the node with the specified ID.

SCE CONFIDENTIAL

findNodeBySid

Returns the node with the specified SID.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class VisualScene {
                    virtual const Node *findNodeBySid(
                        std::string sid
                    )=0 const;
                }
            }
        }
    }
}
```

Arguments

sid SID

Return Values

[Node](#). NULL if not found.

Description

This returns the node with the specified SID.

SCE CONFIDENTIAL

get rootNode

Returns the root node.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class VisualScene {
                    virtual Node *get rootNode(void)=0;
                }
            }
        }
    }
}
```

Return Values

Root node

Description

This returns the root node.

SCE CONFIDENTIAL

get rootNode

Returns the root node.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class VisualScene {
                    virtual const Node *get rootNode(void)=0
                const;
            }
        }
    }
}
```

Return Values

Root node

Description

This returns the root node.

sce::SampleUtil::Graphics::Collada::InstanceVisualScene

Summary

sce::SampleUtil::Graphics::Collada::InstanceVisualScene

Instance of a scene.

Definition

```
#include <collada_scene.h>
class InstanceVisualScene : public sce::SampleUtil::Resource {};
```

Description

This is the class to indicate the instance of a scene.

Methods Summary

Methods	Description
get rootNode	Returns the root node.
get rootNode	Returns the root node.
draw	Draws a scene.
apply Parameter Reference	Changes the reference destination of a shader parameter.
add Effect	Adds an effect.
find Node By Id	Returns the node with the specified ID.
find Node By Sid	Returns the node with the specified SID.
find Node By Id	Returns the node with the specified ID.
find Node By Sid	Returns the node with the specified SID.
get Visual Scene	Returns ViusalScene which was the source.

Public Instance Methods

get rootNode

Returns the root node.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class InstanceVisualScene {
                    virtual Node *get rootNode(void)=0;
                }
            }
        }
    }
}
```

Return Values

Root node

Description

This returns the root node.

SCE CONFIDENTIAL

get rootNode

Returns the root node.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class InstanceVisualScene {
                    virtual const Node *get rootNode(void)=0
                const;
            }
        }
    }
}
```

Return Values

Root node

Description

This returns the root node.

SCE CONFIDENTIAL

draw

Draws a scene.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class InstanceVisualScene {
                    virtual void draw(
                        GraphicsContext *graphicsContext,
                        sce::Vectormath::Simd::Aos::Matrix4_arg matrix =
                        sce::Vectormath::Simd::Aos::Matrix4::identity(),
                        uint82_t renderingStage = 0
                        )=0 const;
                }
            }
        }
    }
}
```

Arguments

<i>graphicsContext</i>	Pointer to a graphics context
<i>matrix</i>	When drawing a node with Mesh , the product of the scene local World matrix of a node and this value is set in the shader parameter.
<i>renderingStage</i>	Rendering stage number. This draws only EffectInstance whose rendering stage matches this number. The rendering stage number of EffectInstance included in a collada file is initialized by 0.

Return Values

None

Description

This draws a scene.

SCE CONFIDENTIAL

applyParameterReference

Changes the reference destination of a shader parameter.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class InstanceVisualScene {
                    virtual void applyParameterReference(
                        ShaderParameterReference
                        *reference
                    )=0;
                }
            }
        }
    }
}
```

Arguments

reference The reference destination of a shader parameter

Return Values

None

Description

This changes the reference destination of a parameter used for shader drawing in a scene. `getParamValue()` of *reference* is called to all parameters, and the reference destination of a parameter to which a value other than NULL was returned is changed.

SCE CONFIDENTIAL

addEffect

Adds an effect.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class InstanceVisualScene {
                    virtual int addEffect(
                        const Graphics::Effect::EffectData*
                        *effect,
                        int renderingStage
                    )=0;
                };
            };
        };
    };
}
```

Arguments

<i>effect</i>	The effect to be added
<i>renderingStage</i>	Rendering stage to draw

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This adds the effect specified in the mesh in a scene. This is used for drawing only when renderingStage specified by draw matches.

SCE CONFIDENTIAL

findNodeById

Returns the node with the specified ID.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class InstanceVisualScene {
                    virtual Node *findNodeById(
                        std::string id
                    )=0;
                }
            }
        }
    }
}
```

Arguments

id ID of a node

Return Values

[Node](#)

Description

This returns the node with the specified ID. This returns NULL if not found.

SCE CONFIDENTIAL

findNodeBySid

Returns the node with the specified SID.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class InstanceVisualScene {
                    virtual Node *findNodeBySid(
                        std::string sid
                    )=0;
                }
            }
        }
    }
}
```

Arguments

sid SID of a node

Return Values

[Node](#)

Description

This returns the node with the specified SID. This returns NULL if not found.

SCE CONFIDENTIAL

findNodeById

Returns the node with the specified ID.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class InstanceVisualScene {
                    virtual const Node *findNodeById(
                        std::string id
                    )=0 const;
                }
            }
        }
    }
}
```

Arguments

id ID of a node

Return Values

[Node](#)

Description

This returns the node with the specified ID. This returns NULL if not found.

SCE CONFIDENTIAL

findNodeBySid

Returns the node with the specified SID.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class InstanceVisualScene {
                    virtual const Node *findNodeBySid(
                        std::string sid
                    )=0 const;
                }
            }
        }
    }
}
```

Arguments

sid SID of a node

Return Values

[Node](#)

Description

This returns the node with the specified SID. This returns NULL if not found.

SCE CONFIDENTIAL

getVisualScene

Returns ViusalScene which was the source.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class InstanceVisualScene {
                    virtual const VisualScene*
*getVisualScene(void)=0 const;
                };
            }
        }
    }
}
```

Return Values

[VisualScene](#)

Description

This returns ViusalScene which was the source.

sce::SampleUtil::Graphics::Collada::AnimationPlayer

Summary

sce::SampleUtil::Graphics::Collada::AnimationPlayer

[Animation](#) player.

Definition

```
#include <collada_scene.h>
class AnimationPlayer : public sce::SampleUtil::Resource {};
```

Description

This transfers the node of [InstanceVisualScene](#) according to the data of [Animation](#) and animate it.

Methods Summary

Methods	Description
setTime	Moves the node of a target scene to the position of the specified time.
 setFrame	Moves the node of a target scene to the position of the specified frame.
getAnimation	Returns animation data.

Public Instance Methods

setTime

Moves the node of a target scene to the position of the specified time.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class AnimationPlayer {
                    virtual void setTime(
                        float time
                    )=0 const;
                }
            }
        }
    }
}
```

Arguments

time [Animation](#) time

Return Values

None

Description

This changes the transform of the node of instanceVisualScene specified at the time of initialization to the value of the specified time of animation data.

If the time is out of range, the nearest valid time will be used.

SCE CONFIDENTIAL

setFrame

Moves the node of a target scene to the position of the specified frame.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class AnimationPlayer {
                    virtual void setFrame(
                        uint32_t frame
                    )=0 const;
                }
            }
        }
    }
}
```

Arguments

frame [Animation](#) frame number

Return Values

None

Description

This changes the transform of the node of instanceVisualScene specified at the time of initialization to the value of the specified frame of animation data.

If the time is out of range, the nearest valid frame will be used. This function is synonymous with setTime (starting time + frame period * frame).

SCE CONFIDENTIAL

getAnimation

Returns animation data.

Definition

```
#include <collada_scene.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Collada {
                class AnimationPlayer {
                    virtual const Animation*
*getAnimation(void)=0 const;
                };
            };
        };
    };
}
```

Return Values

Pointer to [Animation](#). NULL if not found.

Description

This returns the animation data used at the time of initialization.

SCE CONFIDENTIAL

sce::SampleUtil::Graphics::BlendInfo

000004892117

Summary

sce::SampleUtil::Graphics::BlendInfo

The description about blending and masking to the color surface of a shader result.

Definition

```
#include <constant.h>
struct BlendInfo { };
```

Description

This is the description about blending and masking to the color surface of a shader result.

Fields

Public Instance Fields

uint8_t <i>colorMask</i>	The bit field mask using the value of SceGxmColorMask.
uint8_t <i>colorFunc</i>	Color blend function (From SceGxmBlendFunc)
uint8_t <i>alphaFunc</i>	Alpha blend function (From SceGxmBlendFunc)
uint8_t <i>colorSrc</i>	Source color blend coefficient (From SceGxmBlendFactor)
uint8_t <i>colorDst</i>	Destination color blend coefficient (From SceGxmBlendFactor)
uint8_t <i>alphaSrc</i>	Source alpha blend coefficient (From SceGxmBlendFactor)
uint8_t <i>alphaDst</i>	Destination alpha blend coefficient (From SceGxmBlendFactor)

Methods Summary

Methods	Description
initializeAsNoBlend	Initializes by settings without blending.
initializeAsAlphaBlend	Initializes by settings with alpha blending.
initializeAsAddBlend	Initializes by settings with additive blending.

Public Instance Methods

initializeAsNoBlend

Initializes by settings without blending.

Definition

```
#include <constant.h>
inline void initializeAsNoBlend(void);
```

Return Values

None

Description

This initializes by settings without blending.

SCE CONFIDENTIAL

initializeAsAlphaBlend

Initializes by settings with alpha blending.

Definition

```
#include <constant.h>
inline void initializeAsAlphaBlend(void);
```

Return Values

None

Description

This initializes by settings with alpha blending.

SCE CONFIDENTIAL

initializeAsAddBlend

Initializes by settings with additive blending.

Definition

```
#include <constant.h>
inline void initializeAsAddBlend(void);
```

Return Values

None

Description

This initializes by settings with additive blending.

sce::SampleUtil::Graphics::GraphicsContextOption

Summary

sce::SampleUtil::Graphics::GraphicsContextOption

Structure for initializing [GraphicsContext](#).

Definition

```
#include <context.h>
struct GraphicsContextOption { };
```

Description

This is the structure for initializing [GraphicsContext](#). This is used by specifying it to the argument "option" of GraphicsContext::initialize().

Fields

Public Instance Fields

MultisampleMode	The settings of MSAA of the main render target displayed on the display (Default: kMultisampleNone)
<i>multisampleMode</i>	
uint32_t numFrameBuffer	Number of ring buffers for the main render target displayed on the display (Default: 3)

Methods Summary

Methods	Description
GraphicsContextOption	Constructor.

Constructors and Destructors

GraphicsContextOption

Constructor.

Definition

```
#include <context.h>
inline GraphicsContextOption(void);
```

Return Values

None

Description

This is a constructor.

sce::SampleUtil::Graphics::GraphicsContext

Summary

sce::SampleUtil::Graphics::GraphicsContext

[Graphics](#) context.

Definition

```
#include <context.h>
class GraphicsContext : public sce::SampleUtil::Graphics::GraphicsLoader {};
```

Description

Manages the data required for drawing of graphics and sets the state.

Methods Summary

Methods	Description
clearRenderTarget	Clears the render targets that are currently set.
isDisplaymodeSupported	Check display mode support.
setDisplaymode	Change the display mode.
getNextRenderTarget	Gets the render target of the frame buffer to be drawn next.
getDepthStencilSurface	Gets the depth and stencil target for frame buffer.
beginScene	Starts drawing on the render target.
endScene	Terminates drawing on the render target.
flip	Flip processing of a frame buffer.
setDepthFunc	Sets the comparison function of depth values.
setDepthWriteEnable	Enables or disables depth writing.
setFillMode	Sets the fill mode.
setCullMode	Sets the culling mode.
setVertexProgram	Sets a vertex program.
setFragmentProgram	Sets a fragment program.
setVertexBuffer	Sets a vertex buffer.
draw	Drawing.
setUniformBuffer	Sets a uniform buffer.
setTexture	Sets a texture.
reserveDefaultUniformBuffer	Allocates the area for the default uniform buffer.
setViewPort	Sets a view port.
setRegionClip	Sets a region clip.
setStencilFunc	Sets a stencil function.
getCurrentRenderTarget	Gets the current render target.
getCurrentDepthStencilSurface	Gets the current depth and stencil targets.
setLineWidth	Change width of lines and points.
saveBackBufferAsBmp	Saves a back buffer in a BMP file.

Enumerated Types

FrameBufferSide

Frame buffer side.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                enum FrameBufferSide {
                    kFrameBufferSideLeft = 0,
                    kFrameBufferSideRight = 1
                };
            }
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kFrameBufferSideLeft	0	Left side.
kFrameBufferSideRight	1	Right side.

Description

Used for the frame buffer side when Stereoscopic 3D is enabled. When disabled, only LEFT will be displayed on the display.

DisplayMode

Display mode.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                enum DisplayMode {
                    kDisplayModeNormal,
                    kDisplayModeS3dView
                };
            }
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kDisplayModeNormal	N/A	The normal display mode that only shows one side.
kDisplayModeS3dView	N/A	Stereoscopic 3D mode. Displays both the left side and the right side.

Description

This is the display mode. The normal display mode and Stereoscopic 3D display mode are supported.

Public Instance Methods

clearRenderTarget

Clears the render targets that are currently set.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual void clearRenderTarget(
                    uint32_t color
                )=0;
            }
        }
    }
}
```

Arguments

color Color used for filling

Return Values

None

Description

This fills the current render targets with the specified color.

SCE CONFIDENTIAL

isDisplaymodeSupported

Check display mode support.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual bool isDisplaymodeSupported(
                    DisplayMode mode
                )=0;
            }
        }
    }
}
```

Arguments

mode Display mode

Return Values

True if supported, false if not supported.

Description

This checks if the specified display mode is supported by the display.

SCE CONFIDENTIAL

setDisplaymode

Change the display mode.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual int setDisplaymode(
                    DisplayMode mode
                )=0;
            }
        }
    }
}
```

Arguments

mode Display mode

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This changes the display mode. An error will return if the specified mode is not supported by the display. The initial value for the display mode is kDisplayModeNormal.

SCE CONFIDENTIAL

getNextRenderTarget

Gets the render target of the frame buffer to be drawn next.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual RenderTarget *getNextRenderTarget(
                    FrameBufferSide size =
kFrameBufferSideLeft
                )=0;
            }
        }
    }
}
```

Arguments

`size` Frame buffer side (default: `kFrameBufferSideLeft`)

Return Values

Pointer to a render target

Description

This gets the render target of the frame buffer to be drawn next.

SCE CONFIDENTIAL

getDepthStencilSurface

Gets the depth and stencil target for frame buffer.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual DepthStencilSurface
*getDepthStencilSurface(void)=0;
            }
        }
    }
}
```

Return Values

Pointer to a depth and stencil target

Description

This gets the depth and stencil target for frame buffer.

SCE CONFIDENTIAL

beginScene

Starts drawing on the render target.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual int beginScene(
                    RenderTarget *renderTarget,
                    DepthStencilSurface *depthStencilSurface
                )=0;
            }
        }
    }
}
```

Arguments

<i>renderTarget</i>	RenderTarget
<i>depthStencilSurface</i>	Depth stencil target

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This starts drawing on the render target.

SCE CONFIDENTIAL

endScene

Terminates drawing on the render target.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual int endScene(void)=0;
            }
        }
    }
}
```

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This terminates drawing on the render target.

SCE CONFIDENTIAL

flip

Flip processing of a frame buffer.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual int flip(
                    uint32_t numVSync
                )=0;
            }
        }
    }
}
```

Arguments

numVSync Number of VSYNCs to wait until flip

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This executes flip processing of a frame buffer

SCE CONFIDENTIAL

setDepthFunc

Sets the comparison function of depth values.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual void setDepthFunc(
                    DepthFunc depthFunc
                )=0;
            }
        }
    }
}
```

Arguments

depthFunc Comparison function of depth values

Return Values

None

Description

This sets the comparison function of depth values.

SCE CONFIDENTIAL

setDepthWriteEnable

Enables or disables depth writing.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual void setDepthWriteEnable(
                    bool enable
                )=0;
            }
        }
    }
}
```

Arguments

enable Enables or disables. Enabled if it is true.

Return Values

None

Description

This enables or disables depth writing.

SCE CONFIDENTIAL

setFillMode

Sets the fill mode.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual void setFillMode(
                    bool isFill
                )=0;
            }
        }
    }
}
```

Arguments

isFill Enables or disables. Enabled if it is true.

Return Values

None

Description

This sets the fill mode.

SCE CONFIDENTIAL

setCullMode

Sets the culling mode.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual void setCullMode(
                    CullMode cullMode
                )=0;
            }
        }
    }
}
```

Arguments

cullMode Culling mode

Return Values

None

Description

This sets the culling mode.

SCE CONFIDENTIAL

setVertexProgram

Sets a vertex program.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual int setVertexProgram(
                    VertexProgram *vertexProgram
                )=0;
            }
        }
    }
}
```

Arguments

vertexProgram Vertex program

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This sets a vertex program.

SCE CONFIDENTIAL

setFragmentProgram

Sets a fragment program.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual int setFragmentProgram(
                    FragmentProgram *fragmentProgram
                )=0;
            }
        }
    }
}
```

Arguments

fragmentProgram Fragment program

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This sets a fragment program.

SCE CONFIDENTIAL

setVertexBuffer

Sets a vertex buffer.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual int setVertexBuffer(
                    VertexBuffer *vertexBuffer,
                    uint32_t streamIndex
                )=0;
            }
        }
    }
}
```

Arguments

<i>vertexBuffer</i>	Vertex buffer
<i>streamIndex</i>	Index of buffers

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This sets a vertex buffer.

SCE CONFIDENTIAL

draw

Drawing.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual int draw(
                    Primitive primitive,
                    IndexBuffer *indexBuffer,
                    uint32_t count
                )=0;
            }
        }
    }
}
```

Arguments

<i>primitive</i>	Drawing primitive
<i>indexBuffer</i>	Index buffer.
<i>count</i>	Number of indices

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This Draws.

SCE CONFIDENTIAL

setUniformBuffer

Sets a uniform buffer.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual int setUniformBuffer(
                    ProgramType programType,
                    UniformBuffer *buffer,
                    uint32_t bufferIndex
                )=0;
            }
        }
    }
}
```

Arguments

<i>programType</i>	Type of a shader program
<i>buffer</i>	Uniform buffer
<i>bufferIndex</i>	Index of uniform buffers

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This sets a uniform buffer.

SCE CONFIDENTIAL

setTexture

Sets a texture.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual int setTexture(
                    ProgramType programType,
                    Texture *texture,
                    uint32_t index
                )=0;
            }
        }
    }
}
```

Arguments

<i>programType</i>	Type of a shader program
<i>texture</i>	Texture
<i>index</i>	Index of textures

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This sets a texture.

SCE CONFIDENTIAL

reserveDefaultUniformBuffer

Allocates the area for the default uniform buffer.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual void *reserveDefaultUniformBuffer(
                    ProgramType programType
                )=0;
            }
        }
    }
}
```

Arguments

programType Type of a shader program

Return Values

Pointer to an area

Description

This allocates the area for the default uniform buffer. Call this function after setting a shader program.

setViewPort

Sets a view port.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual void setViewPort(
                    uint32_t topLeftX,
                    uint32_t topLeftY,
                    uint32_t width,
                    uint32_t height,
                    float minDepth,
                    float maxDepth
                )=0;
            }
        }
    }
}
```

Arguments

<i>topLeftX</i>	The x coordinate at the upper left of a view port
<i>topLeftY</i>	The y coordinate at the upper left of a view port
<i>width</i>	Width of a view port
<i>height</i>	Height of a view port
<i>minDepth</i>	The minimum depth of a view port (0 to 1)
<i>maxDepth</i>	The maximum depth of a view port (0 to 1)

Return Values

None

Description

This sets a view port. In PlayStation®Vita, the coordinate for clipping is aligned to the granularity of the tile. This alignment will be performed so that the view port will be in an area inside the area specified with the argument.

SCE CONFIDENTIAL

setRegionClip

Sets a region clip.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual void setRegionClip(
                    uint32_t topLeftX,
                    uint32_t topLeftY,
                    uint32_t width,
                    uint32_t height
                )=0;
            }
        }
    }
}
```

Arguments

<i>topLeftX</i>	The x coordinate at the upper left of a clip area.
<i>topLeftY</i>	The y coordinate at the upper left of a clip area.
<i>width</i>	Width of a clip region
<i>height</i>	Height of a clip area

Return Values

None

Description

This sets a region clip. In PlayStation®Vita, the coordinate for clipping is aligned to the granularity of the tile. This alignment will be performed so that the view port will be in an area inside the area specified with the argument.

SCE CONFIDENTIAL

setStencilFunc

Sets a stencil function.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual void setStencilFunc(
                    uint8_t compareMask,
                    uint8_t writeMask,
                    StencilFunc frontFaceFunc,
                    StencilOp frontFaceStencilFail,
                    StencilOp frontFaceDepthFail,
                    StencilOp frontFaceDepthPass,
                    StencilFunc backFaceFunc,
                    StencilOp backFaceStencilFail,
                    StencilOp backFaceDepthFail,
                    StencilOp backFaceDepthPass,
                    uint8_t stencilRef
                )=0;
            }
        }
    }
}
```

Arguments

<i>compareMask</i>	Bit mask used for comparison of stencil buffers. An AND operation is executed for the current stencil value with this value before conducting a test.
<i>writeMask</i>	Mask on a bit basis applied to stencil values after stencil operations
<i>frontFaceFunc</i>	Stencil comparison function for the front
<i>frontFaceStencilFail</i>	Stencil operation executed when a stencil test for the front fails
<i>frontFaceDepthFail</i>	Stencil operation executed when a depth test for the front fails
<i>frontFaceDepthPass</i>	Stencil operation executed when a depth test for the front is passed
<i>backFaceFunc</i>	Stencil comparison function for the back
<i>backFaceStencilFail</i>	Stencil operation executed when a stencil test for the back fails
<i>backFaceDepthFail</i>	Stencil operation executed when a depth test for the back fails
<i>backFaceDepthPass</i>	Stencil operation executed when a depth test for the back is passed
<i>stencilRef</i>	Stencil reference value

Return Values

None

Description

This sets a stencil function.

SCE CONFIDENTIAL

getCurrentRenderTarget

Gets the current render target.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual RenderTarget
            *getCurrentRenderTarget(void)=0 const;
        }
    }
}
```

Return Values

Current render target

Description

This gets the current render target.

getCurrentDepthStencilSurface

Gets the current depth and stencil targets.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual DepthStencilSurface
*getCurrentDepthStencilSurface(void)=0 const;
            }
        }
    }
}
```

Return Values

Current depth and stencil targets

Description

This gets the current depth and stencil targets.

SCE CONFIDENTIAL

setLineWidth

Change width of lines and points.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual void setLineWidth(
                    uint32_t width
                )=0;
            }
        }
    }
}
```

Arguments

width Line and dot width

Return Values

None

Description

This changes the pixel unit for the width of lines and points.

SCE CONFIDENTIAL

saveBackBufferAsBmp

Saves a back buffer in a BMP file.

Definition

```
#include <context.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsContext {
                virtual int saveBackBufferAsBmp(
                    const char *path
                )=0;
            }
        }
    }
}
```

Arguments

path Path of a BMP file

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This saves a back buffer in a BMP file.

SCE CONFIDENTIAL

sce::SampleUtil::Graphics::Effect

000004892117

Summary

sce::SampleUtil::Graphics::Effect

Name space associated with the effect.

Definition

```
namespace Effect {}
```

Description

This is the name space associated with the effect.

Inner Classes, Structures, and Namespaces

Item	Description
sce::SampleUtil::Graphics::Effect::ParameterValue	The value of a shader parameter.
sce::SampleUtil::Graphics::Effect::EffectParameter	Uniform parameter with a value in an effect.
sce::SampleUtil::Graphics::Effect::EffectShader	The shader information in an effect.
sce::SampleUtil::Graphics::Effect::EffectData	Effect class.
sce::SampleUtil::Graphics::Effect::EffectInstance	Instance of an effect.

sce::SampleUtil::Graphics::Effect::ParameterValue

Summary

sce::SampleUtil::Graphics::Effect::ParameterValue

The value of a shader parameter.

Definition

```
#include <effect.h>
class ParameterValue {};
```

Description

This is the value of a shader parameter.

Methods Summary

Methods	Description
ParameterValue	Constructor.
~ParameterValue	Destructor.
initializeAsArray	Initializes as array data.
initializeAsTexture	Initializes as texture data.
setArrayDataF	Sets a value in array data.
setArrayData	Sets a value in array data.
setTexture	Set texture data.
finalize	Termination processing.
getCategory	Returns the category of a value.
getType	Returns the type of a value.
getData	Gets the pointer to array data.
getData	Gets the pointer to array data.
getTexture	Gets texture data.
copyFrom	Generates a copy.

Constructors and Destructors

ParameterValue

Constructor.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class ParameterValue {
                    ParameterValue(void);
                }
            }
        }
    }
}
```

Return Values

None

Description

This is a constructor.

SCE CONFIDENTIAL

~ParameterValue

Destructor.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class ParameterValue {
                    ~ParameterValue(void);
                }
            }
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Public Instance Methods

initializeAsArray

Initializes as array data.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class ParameterValue {
                    int initializeAsArray(
                        ParameterType type,
                        uint32_t count,
                        const void *initData
                    );
                }
            }
        }
    }
}
```

Arguments

<i>type</i>	Type of an element of an array
<i>count</i>	Number of elements
<i>initData</i>	The initial value. It is ignored in the case of NULL.

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This initializes as array data.

SCE CONFIDENTIAL

initializeAsTexture

Initializes as texture data.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class ParameterValue {
                    int initializeAsTexture(
                        Texture *texture
                    );
                }
            }
        }
    }
}
```

Arguments

texture [Texture](#)

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This initializes as texture data.

SCE CONFIDENTIAL

setArrayDataF

Sets a value in array data.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class ParameterValue {
                    int setArrayDataF(
                        const float *value,
                        uint32_t componentCount = 0
                    );
                }
            }
        }
    }
}
```

Arguments

<i>value</i>	Value to set
<i>componentCount</i>	Number of elements to set. If 0 is specified, the number of elements specified at the time of initialization will be set.

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This sets a value in array data. The value of float is cast in the type of an array element appropriately and set.

SCE CONFIDENTIAL

setArrayData

Sets a value in array data.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class ParameterValue {
                    int setArrayData(
                        const void *value,
                        uint32_t numBytes = 0
                    );
                };
            }
        }
    }
}
```

Arguments

<i>value</i>	Value to set
<i>numBytes</i>	Number of bytes copied in an array. If 0 is specified, the size allocated at the time of initialization will be copied.

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This sets a value in array data. The value of *value* is copied as it is.

SCE CONFIDENTIAL

setTexture

Set texture data.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class ParameterValue {
                    int setTexture(
                        Texture *texture
                    );
                }
            }
        }
    }
}
```

Arguments

texture [Texture](#)

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This sets texture data.

SCE CONFIDENTIAL

finalize

Termination processing.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class ParameterValue {
                    int finalize(void);
                }
            }
        }
    }
}
```

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This executes termination processing to unload internal resources.

SCE CONFIDENTIAL

getCategory

Returns the category of a value.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class ParameterValue {
                    inline ParameterCategory
getCategory(void) const;
                }
            }
        }
    }
}
```

Return Values

Category of a value

Description

This returns PARAMETER_CATEGORY_UNIFORM in the case of array data and PARAMETER_CATEGORY_SAMPLER in the case of texture data.

SCE CONFIDENTIAL

get~~Type~~

Returns the type of a value.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class ParameterValue {
                    inline ParameterType getType(void) const;
                }
            }
        }
    }
}
```

Return Values

Type of a value

Description

The type of an element in the case of array data This returns PARAMETER_TYPE_SAMPLER in the case of a texture.

SCE CONFIDENTIAL

getData

Gets the pointer to array data.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class ParameterValue {
                    inline void *getData(void);
                }
            }
        }
    }
}
```

Return Values

Pointer to array data

Description

This gets the pointer to array data.

SCE CONFIDENTIAL

getData

Gets the pointer to array data.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class ParameterValue {
                    inline const void *getData(void) const;
                }
            }
        }
    }
}
```

Return Values

Pointer to array data

Description

This gets the pointer to array data.

SCE CONFIDENTIAL

getTexture

Gets texture data.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class ParameterValue {
                    inline Texture *getTexture(void) const;
                }
            }
        }
    }
}
```

Return Values

[Texture](#) Data

Description

This gets texture data.

SCE CONFIDENTIAL

copyFrom

Generates a copy.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class ParameterValue {
                    int copyFrom(
                        const ParameterValue *from
                    );
                }
            }
        }
    }
}
```

Arguments

from Copy source

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Failure

Description

This generates an instance with the same contents as from.

sce::SampleUtil::Graphics::Effect::EffectParameter

Summary

sce::SampleUtil::Graphics::Effect::EffectParameter

Uniform parameter with a value in an effect.

Definition

```
#include <effect.h>
class EffectParameter {};
```

Description

This is a uniform parameter with a value in an effect.

Methods Summary

Methods	Description
EffectParameter	Constructor.
initialize	Initialization.
getParameter	Gets a shader parameter.
getValue	Gets a parameter value.
getValue	Gets a parameter value.
setToUniformBuffer	Copies a value to the inside of a uniform buffer.
finalize	Termination processing.
~EffectParameter	Destructor.
copyFrom	Generates a copy.
getProgramType	Gets a program type.

Constructors and Destructors

EffectParameter

Constructor.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectParameter {
                    EffectParameter(void);
                }
            }
        }
    }
}
```

Return Values

None

Description

This is a constructor.

SCE CONFIDENTIAL

~EffectParameter

Destructor.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectParameter {
                    ~EffectParameter(void);
                }
            }
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Public Instance Methods

initialize

Initialization.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectParameter {
                    int initialize(
                        const Parameter *param,
                        ProgramType programType
                    );
                }
            }
        }
    }
}
```

Arguments

<i>param</i>	Shader parameter
<i>programType</i>	Type of the program of this parameter

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This executes initialization.

SCE CONFIDENTIAL

getParameter

Gets a shader parameter.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectParameter {
                    inline const Parameter *getParameter(void)
const;
```

Return Values

Shader parameter

Description

This gets a shader parameter.

SCE CONFIDENTIAL

getValue

Gets a parameter value.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectParameter {
                    inline ParameterValue *getValue(void);
                }
            }
        }
    }
}
```

Return Values

[Parameter](#) value

Description

This gets a parameter value.

SCE CONFIDENTIAL

getValue

Gets a parameter value.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectParameter {
                    inline const ParameterValue
                *getValue(void) const;
                }
            }
        }
    }
}
```

Return Values

[Parameter](#) value

Description

This gets a parameter value.

SCE CONFIDENTIAL

setToUniformBuffer

Copies a value to the inside of a uniform buffer.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectParameter {
                    int setToUniformBuffer(
                        GraphicsContext *context,
                        void *buffer
                    );
                }
            }
        }
    }
}
```

Arguments

context [Graphics](#) context
buffer Uniform buffer

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Failure

Description

The value is copied to the offset position of this parameter within a uniform buffer.

SCE CONFIDENTIAL

finalize

Termination processing.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectParameter {
                    int finalize(void);
                }
            }
        }
    }
}
```

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This executes termination processing to unload internal resources.

SCE CONFIDENTIAL

copyFrom

Generates a copy.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectParameter {
                    int copyFrom(
                        const EffectParameter *from
                    );
                }
            }
        }
    }
}
```

Arguments

from [Parameter](#) to be the copy source

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Failure

Description

This generates a copy.

SCE CONFIDENTIAL

getProgramType

Gets a program type.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectParameter {
                    ProgramType getProgramType(void) const;
                }
            }
        }
    }
}
```

Return Values

[Program](#) type

Description

This returns the type of the program to which this parameter belongs.

sce::SampleUtil::Graphics::Effect::EffectShader

Summary

sce::SampleUtil::Graphics::Effect::EffectShader

The shader information in an effect.

Definition

```
#include <effect.h>
class EffectShader {};
```

Description

This is the shader information in an effect.

Methods Summary

Methods	Description
initialize	Initialization.
getNumParams	Gets the number of parameters.
getParamIndex	Gets the index of parameters.
getParamByIndex	Gets a parameter.
getParamByName	Gets a parameter.
getParamByIndex	Gets a parameter.
getParamByName	Gets a parameter.
getProgramId	Gets a program ID.
finalize	Termination processing.
EffectShader	Constructor.
~EffectShader	Destructor.
copyFrom	Generates a copy.

Constructors and Destructors

EffectShader

Constructor.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectShader {
                    EffectShader(void);
                }
            }
        }
    }
}
```

Return Values

None

Description

This is a constructor.

SCE CONFIDENTIAL

~EffectShader

Destructor.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectShader {
                    ~EffectShader(void);
                }
            }
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Public Instance Methods

initialize

Initialization.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectShader {
                    int initialize(
                        const Program *programId,
                        ProgramType programType
                    );
                };
            }
        }
    }
}
```

Arguments

programId ID of a shader program
programType [Program](#) type

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This initializes the shader information in an effect.

SCE CONFIDENTIAL

getNumParams

Gets the number of parameters.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectShader {
                    uint32_t getNumParams(void) const;
                }
            }
        }
    }
}
```

Return Values

Number of parameters

Description

This gets the number of parameters.

SCE CONFIDENTIAL

getParamIndex

Gets the index of parameters.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectShader {
                    int getParamIndex(
                        std::string name
                    ) const;
                }
            }
        }
    }
}
```

Arguments

name [Parameter](#) name

Return Values

Index of parameters. -1 if not found.

Description

This returns the index of the parameters of the specified name.

SCE CONFIDENTIAL

getParamByIndex

Gets a parameter.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectShader {
                    EffectParameter *getParamByIndex(
                        uint32_t index
                    );
                }
            }
        }
    }
}
```

Arguments

index Index of parameters

Return Values

[Parameter](#). NULL if index is greater than the number of parameters.

Description

This gets a parameter.

SCE CONFIDENTIAL

getParamByName

Gets a parameter.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectShader {
                    EffectParameter *getParamByName(
                        std::string name
                    );
                }
            }
        }
    }
}
```

Arguments

name [Parameter](#) name

Return Values

[Parameter](#). NULL if not found.

Description

This gets a parameter.

SCE CONFIDENTIAL

getParamByIndex

Gets a parameter.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectShader {
                    const EffectParameter *getParamByIndex(
                        uint32_t index
                    ) const;
                }
            }
        }
    }
}
```

Arguments

index Index of parameters

Return Values

[Parameter](#). NULL if index is greater than the number of parameters.

Description

This gets a parameter.

SCE CONFIDENTIAL

getParamByName

Gets a parameter.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectShader {
                    const EffectParameter *getParamByName(
                        std::string name
                    ) const;
                }
            }
        }
    }
}
```

Arguments

name [Parameter](#) *name*

Return Values

[Parameter](#). NULL if not found.

Description

This gets a parameter.

SCE CONFIDENTIAL

getProgramId

Gets a program ID.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectShader {
                    inline const Program *getProgramId(void)
const;
```

Return Values

[Program](#) ID

Description

This gets a program ID.

SCE CONFIDENTIAL

finalize

Termination processing.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectShader {
                    int finalize(void);
                }
            }
        }
    }
}
```

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This executes termination processing to unload internal resources.

SCE CONFIDENTIAL

copyFrom

Generates a copy.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectShader {
                    int copyFrom(
                        const EffectShader *from
                    );
                }
            }
        }
    }
}
```

Arguments

from Copy source

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Failure

Description

This generates an instance with the same contents as from.

SCE CONFIDENTIAL

sce::SampleUtil::Graphics::Effect::EffectData

Summary

sce::SampleUtil::Graphics::Effect::EffectData

[Effect](#) class.

Definition

```
#include <effect.h>
class EffectData {};
```

Description

This is the effect class.

Methods Summary

Methods	Description
initialize	Initialization.
finalize	Termination processing.
getVertexShader	Returns the information of a vertex shader.
getFragmentShader	Returns the information of a fragment shader.
getVertexShader	Returns the information of a vertex shader.
getFragmentShader	Returns the information of a fragment shader.
getNumAttributeParams	Returns the number of the attribute parameters of a vertex shader.
getAttributeParam	Gets the attribute parameters of a vertex shader.
copyFrom	Generates a copy.
EffectData	Constructor.
~EffectData	Destructor.

Constructors and Destructors

EffectData

Constructor.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectData {
                    EffectData(void);
                }
            }
        }
    }
}
```

Return Values

None

Description

This is a constructor.

SCE CONFIDENTIAL

~EffectData

Destructor.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectData {
                    ~EffectData(void);
                }
            }
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Public Instance Methods

initialize

Initialization.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectData {
                    int initialize(
                        const Program *vertexProgramId,
                        const Program *fragmentProgramId,
                        BlendInfo *blendInfo
                    );
                }
            }
        }
    }
}
```

Arguments

<i>vertexProgramId</i>	ID of a vertex program
<i>fragmentProgramId</i>	ID of a fragment program
<i>blendInfo</i>	Blend information. If not blended, specify NULL.

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This executes initialization.

SCE CONFIDENTIAL

finalize

Termination processing.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectData {
                    int finalize(void);
                }
            }
        }
    }
}
```

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This executes termination processing to unload internal resources.

SCE CONFIDENTIAL

getVertexShader

Returns the information of a vertex shader.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectData {
                    inline const EffectShader*
                *getVertexShader(void) const;
                }
            }
        }
    }
}
```

Return Values

Information of a vertex shader

Description

This returns the information of a vertex shader.

SCE CONFIDENTIAL

getFragmentShader

Returns the information of a fragment shader.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectData {
                    inline const EffectShader*
*getFragmentShader(void) const;
                }
            }
        }
    }
}
```

Return Values

Information of a fragment shader

Description

This returns the information of a fragment shader.

SCE CONFIDENTIAL

getVertexShader

Returns the information of a vertex shader.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectData {
                    inline EffectShader
*getVertexShader(void);
```

Return Values

Information of a vertex shader

Description

This returns the information of a vertex shader.

SCE CONFIDENTIAL

getFragmentShader

Returns the information of a fragment shader.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectData {
                    inline EffectShader
*getFragmentShader(void);
```

Return Values

Information of a fragment shader

Description

This returns the information of a fragment shader.

SCE CONFIDENTIAL

getNumAttributeParams

Returns the number of the attribute parameters of a vertex shader.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectData {
                    inline uint32_t
getNumAttributeParams(void) const;
                }
            }
        }
    }
}
```

Return Values

Number of the attribute parameters of a vertex shader

Description

This returns the number of the attribute parameters of a vertex shader.

SCE CONFIDENTIAL

getAttributeParam

Gets the attribute parameters of a vertex shader.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectData {
                    inline const Parameter*getAttributeParam(
                        uint32_t index
                    ) const;
                }
            }
        }
    }
}
```

Arguments

index Index of attribute parameters

Return Values

Attribute parameter

Description

This gets the attribute parameters of a vertex shader.

SCE CONFIDENTIAL

copyFrom

Generates a copy.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectData {
                    int copyFrom(
                        const EffectData *from
                    );
                }
            }
        }
    }
}
```

Arguments

from Copy source

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Failure

Description

This generates a new instance with the same contents as from.

SCE CONFIDENTIAL

sce::SampleUtil::Graphics::Effect::Effect tInstance

000004892117

Summary

sce::SampleUtil::Graphics::Effect::EffectInstance

Instance of an effect.

Definition

```
#include <effect.h>
class EffectInstance {};
```

Description

This is the instance of an effect.

Methods Summary

Methods	Description
<u>EffectInstance</u>	Constructor.
<u>~EffectInstance</u>	Destructor.
<u>initialize</u>	Initialization.
<u>finalize</u>	Termination processing.
<u>getEffect</u>	Gets the effect which was the generation source.
<u>setParamValueReferenceByName</u>	Sets the reference of a shader parameter.
<u>apply</u>	Sets an effect in GPU.
<u>cloneFrom</u>	Generates a copy.

Constructors and Destructors

EffectInstance

Constructor.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectInstance {
                    EffectInstance(void);
                }
            }
        }
    }
}
```

Return Values

None

Description

This is a constructor.

SCE CONFIDENTIAL

~EffectInstance

Destructor.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectInstance {
                    virtual ~EffectInstance(void);
                }
            }
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Public Instance Methods

initialize

Initialization.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectInstance {
                    int initialize(
                        GraphicsLoader *graphicsLoader,
                        const EffectData *effect,
                        const VertexAttribute *attrs,
                        uint32_t numAttrs,
                        const VertexStream *streams,
                        uint32_t numStreams
                    );
                };
            }
        }
    }
}
```

Arguments

<i>graphicsLoader</i>	Graphics loader
<i>effect</i>	The effect to be the source
<i>attrs</i>	Array of vertex attributes
<i>numAttrs</i>	Size of an array of vertex attributes
<i>streams</i>	Array of vertex streams
<i>numStreams</i>	Number of vertex streams

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This generates an instance effect. The uniform parameter is initialized to refer to the value of a parameter in the effect which was the generation source.

SCE CONFIDENTIAL

finalize

Termination processing.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectInstance {
                    int finalize(void);
                }
            }
        }
    }
}
```

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This executes termination processing to unload internal resources.

SCE CONFIDENTIAL

getEffect

Gets the effect which was the generation source.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectInstance {
                    inline const EffectData *getEffect(void)
const;
```

Return Values

[Effect](#)

Description

This gets the effect which was the generation source.

SCE CONFIDENTIAL

setParamValueReferenceByName

Sets the reference of a shader parameter.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectInstance {
                    int setParamValueReferenceByName(
                        ProgramType programType,
                        const char *name,
                        const ParameterValue *value
                    );
                }
            }
        }
    }
}
```

Arguments

<i>programType</i>	Type of a shader program
<i>name</i>	Parameter name
<i>value</i>	New reference destination

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This sets the reference destination of a uniform parameter. The value of a parameter is read from the reference destination when the apply function is called. Each parameter refers to the value of the corresponding parameter in the effect which was the generation source at the time of generation.

SCE CONFIDENTIAL

apply

Sets an effect in GPU.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectInstance {
                    int apply(
                        GraphicsContext *context,
                        VertexBuffer *const *vertexBuffer,
                        uint32_t numVertexBuffers
                    );
                }
            }
        }
    }
}
```

Arguments

<i>context</i>	Graphics context
<i>vertexBuffer</i>	Vertex buffer array
<i>numVertexBuffers</i>	Number of vertex buffer arrays

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

Sets the contents of this effect instance and make it drawable. After this function is called, drawing using this effect by the draw call becomes possible.

SCE CONFIDENTIAL

cloneFrom

Generates a copy.

Definition

```
#include <effect.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Effect {
                class EffectInstance {
                    void cloneFrom(
                        const EffectInstance &from
                    );
                }
            }
        }
    }
}
```

Arguments

from Copy source

Return Values

None

Description

This generates a new instance with the same contents as from.

sce::SampleUtil::Graphics::FontParam

000004892117

Summary

sce::SampleUtil::Graphics::FontParam

Structure used to initialize the [Font](#) class.

Definition

```
#include <font.h>
struct FontParam {};
```

Description

This is used as the argument of initialize by initializing with initParam of the [Font](#) class. It is possible to set various operating environments associated with texture generation for the [Font](#) class.

Fields

Public Instance Fields

[FontLanguage](#) *language*

Photo language code.

The initial value is kFontLanguageLatin.

[FontFamily](#) *family*

[Font](#) family code.

The initial value is kFontFamilySansSerif.

[FontWeight](#) *weight*

[Font](#) weight.

The initial value is kFontWeightNormal.

bool *isItalic*

Italic or not.

The initial value is false.

float *fontSizePixelH*

Horizontal font size.

This is specified by pixel value. The initial value is 64.0.

float *fontSizePixelV*

Vertical font size.

This is specified by pixel value. The initial value is 64.0.

float *baselineOffsetY*

Baseline offset value.

Specify the baseline position used when character images are lined up horizontally with the offset value (pixel) from the top edge of the design. If not specified (when the value is 0), the baseline will be automatically determined.

Methods Summary

Methods	Description
setDefaults	Initialize FontParam .

Public Instance Methods

setDefaults

Initialize [FontParam](#).

Definition

```
#include <font.h>
void setDefaults(void);
```

Return Values

None

Description

This is the initialization function of [FontParam](#). This sets the default value in each variable of [FontParam](#). Change each set value after initializing [FontParam](#) with initParam.

SCE CONFIDENTIAL

sce::SampleUtil::Graphics::Font

000004892117

Summary

sce::SampleUtil::Graphics::Font

Class to handle the font texture.

Definition

```
#include <font.h>
class Font : public sce::SampleUtil::Resource {};
```

Description

This generates a texture by specifying a UCS2 code. This has a cache function for the texture generated.

Methods Summary

Methods	Description
cacheCharacters	Cache of a character texture.
clearCache	Clears the cache.

Public Instance Methods

cacheCharacters

Cache of a character texture.

Definition

```
#include <font.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Font {
                virtual int cacheCharacters(
                    const uint16_t *ucs2Charcode
                )=0;
            }
        }
    }
}
```

Arguments

ucs2Charcode Pointer to a UCS2 character code

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid

Description

This creates a character texture by specifying the pointer to UCS2 character code and caches it. It is possible to specify multiple characters. Make sure to always set the termination of a UCS2 character code to NULL.

SCE CONFIDENTIAL

clearCache

Clears the cache.

Definition

```
#include <font.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Font {
                virtual void clearCache(void)=0;
            }
        }
    }
}
```

Return Values

None

Description

This clears the cache of a character texture.

sce::SampleUtil::Graphics::FontLoader

000004892117

Summary

sce::SampleUtil::Graphics::FontLoader

Class that generates fonts.

Definition

```
#include <font.h>
class FontLoader : public sce::SampleUtil::Resource {};
```

Description

This is the class that generates fonts.

Methods Summary

Methods	Description
createFont	Generate font.

Public Instance Methods

createFont

Generate font.

Definition

```
#include <font.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class FontLoader {
                virtual int createFont(
                    Font **outFont,
                    const FontParam *param = NULL,
                    uint32_t numMaxCache = 1024
                )=0;
            }
        }
    }
}
```

Arguments

<i>outFont</i>	Pointer to which the generated font returns
<i>param</i>	Pointer to FontParam . This is initialized by the default value if NULL is specified.
<i>numMaxCache</i>	This is the limit for the cache that can be saved in a font. The limit that can be specified varies depending on the font size.

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid

Description

This generates fonts.

sce::SampleUtil::Graphics::VertexProgr

am

000004892117

Summary

sce::SampleUtil::Graphics::VertexProgram

Vertex program.

Definition

```
#include <loader.h>
class VertexProgram : public sce::SampleUtil::Resource {};
```

Description

This is a vertex program.

SCE CONFIDENTIAL

sce::SampleUtil::Graphics::FragmentProgram

Summary

sce::SampleUtil::Graphics::FragmentProgram

Fragment program.

Definition

```
#include <loader.h>
class FragmentProgram : public sce::SampleUtil::Resource {};
```

Description

This is a fragment program.

sce::SampleUtil::Graphics::VertexStrea

m

000004892117

Summary

sce::SampleUtil::Graphics::VertexStream

Vertex stream.

Definition

```
#include <loader.h>
struct VertexStream {};
```

Description

This is the definition of a vertex stream.

Fields

Public Instance Fields

`uint16_t stride`
`uint16_t indexSource`

Stride of a vertex stream.

Index method. Specify a value of the IndexSource type.

**sce::SampleUtil::Graphics::VertexAttrib
ute**

Summary

sce::SampleUtil::Graphics::VertexAttribute

Structure to indicate the layout of [VertexBuffer](#).

Definition

```
#include <loader.h>
struct VertexAttribute {};
```

Description

This is the structure to indicate the layout of [VertexBuffer](#). This is used when generating [VertexProgram](#).

Fields

Public Instance Fields

<code>uint16_t streamIndex</code>	Index of Vertex streams.
<code>uint16_t offset</code>	Offset in Vertex stream.
<code>AttributeFormat format</code>	Format of the value of an attribute parameter.
<code>uint8_t componentCount</code>	Number of the values of an attribute parameter.
<code>ParameterSemantic semantic</code>	Semantic of an attribute parameter.
<code>uint32_t semanticIndex</code>	Semantic index of attribute parameters.

**sce::SampleUtil::Graphics::GraphicsLo
ader**

000004892117

Summary

sce::SampleUtil::Graphics::GraphicsLoader

Class to manage the resources required for drawing of graphics.

Definition

```
#include <loader.h>
class GraphicsLoader : public sce::SampleUtil::Resource {};
```

Description

This is the class to manage the resources required for drawing of graphics.

Methods Summary

Methods	Description
registerVertexProgram	Registers a vertex program.
registerFragmentProgram	Registers a fragment program.
createVertexProgram	Generates a vertex program.
createFragmentProgram	Generates a fragment program.
cloneVertexProgram	Copies a vertex program.
cloneFragmentProgram	Copies a fragment program.
createIndexBuffer	Create index buffer.
createVertexBuffer	Create vertex buffer.
createUniformBuffer	Create uniform buffer.
createBuffer	Create buffer.
createTexture2dBuffer	Create a two-dimensional texture buffer.
createRenderTarget	Create render target.
createDepthStencilSurface	Create depth and stencil target.
createTextureFromFile	This generates a texture from an image file.
createTextureFromMemory	Create texture.
createTexture	Create texture.

Public Instance Methods

registerVertexProgram

Registers a vertex program.

Definition

```
#include <loader.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsLoader {
                virtual int registerVertexProgram(
                    Program **outProgram,
                    const void *binary,
                    uint32_t binarySize,
                    ShaderFormat shaderFormat =
                    kShaderFormatDefault
                )=0;
            }
        }
    }
}
```

Arguments

<i>outProgram</i>	Pointer to which the generated program returns
<i>binary</i>	Pointer to a vertex program
<i>binarySize</i>	Size (bytes) of the program
<i>shaderFormat</i>	Shader format

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This registers a vertex program.

registerFragmentProgram

Registers a fragment program.

Definition

```
#include <loader.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsLoader {
                virtual int registerFragmentProgram(
                    Program **outProgram,
                    const void *binary,
                    uint32_t binarySize,
                    ShaderFormat shaderFormat =
                    kShaderFormatDefault
                )=0;
            }
        }
    }
}
```

Arguments

<i>outProgram</i>	Pointer to which the generated program returns
<i>binary</i>	Pointer to a fragment program
<i>binarySize</i>	Size (bytes) of the program
<i>shaderFormat</i>	Shader format

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This registers a fragment program.

SCE CONFIDENTIAL

createVertexProgram

Generates a vertex program.

Definition

```
#include <loader.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsLoader {
                virtual int createVertexProgram(
                    VertexProgram **vertexProgram,
                    const Program *programId,
                    const VertexAttribute *attrs,
                    uint32_t numAttrs,
                    const VertexStream *streams,
                    uint32_t numStreams
                )=0;
            }
        }
    }
}
```

Arguments

<i>vertexProgram</i>	Pointer to a vertex program
<i>programId</i>	ID of a vertex program
<i>attrs</i>	Array of attributes
<i>numAttrs</i>	Length of an array of attributes
<i>streams</i>	Array of streams
<i>numStreams</i>	Length of an array of streams

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This generates a vertex program. To delete the generated instance, use [sce::SampleUtil::destroy\(\)](#).

SCE CONFIDENTIAL

createFragmentProgram

Generates a fragment program.

Definition

```
#include <loader.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsLoader {
                virtual int createFragmentProgram(
                    FragmentProgram **fragmentProgram,
                    const Program *programId,
                    const Program *vertexProgramId,
                    const BlendInfo *blendInfo
                )=0;
            }
        }
    }
}
```

Arguments

<i>fragmentProgram</i>	Pointer to a fragment program
<i>programId</i>	ID of a fragment program
<i>vertexProgramId</i>	ID of a vertex program used simultaneously with a fragment program
<i>blendInfo</i>	Blend information. If not blended, NULL is acceptable.

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This generates a fragment program. To delete the generated instance, use [sce::SampleUtil::destroy\(\)](#).

SCE CONFIDENTIAL

cloneVertexProgram

Copies a vertex program.

Definition

```
#include <loader.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsLoader {
                virtual int cloneVertexProgram(
                    VertexProgram **vertexProgram,
                    const VertexProgram *from
                )=0;
            }
        }
    }
}
```

Arguments

<i>vertexProgram</i>	Vertex program of the copy destination
<i>from</i>	Vertex program of the copy source

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This copies a vertex program.

SCE CONFIDENTIAL

cloneFragmentProgram

Copies a fragment program.

Definition

```
#include <loader.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsLoader {
                virtual int cloneFragmentProgram(
                    FragmentProgram **fragmentProgram,
                    const FragmentProgram *from
                )=0;
            }
        }
    }
}
```

Arguments

<i>fragmentProgram</i>	Fragment program of the copy destination
<i>from</i>	Fragment program of the copy source

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This copies a fragment program.

createIndexBuffer

Create index buffer.

Definition

```
#include <loader.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsLoader {
                virtual int createIndexBuffer(
                    IndexBuffer **outIndexBuffer,
                    uint32_t size
                )=0;
            }
        }
    }
}
```

Arguments

outIndexBuffer Pointer to which the generated index buffer returns
size [Buffer](#) size

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This generates an index buffer. To delete the generated instance, use [sce::SampleUtil::destroy\(\)](#).

createVertexBuffer

Create vertex buffer.

Definition

```
#include <loader.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsLoader {
                virtual int createVertexBuffer(
                    VertexBuffer **outVertexBuffer,
                    uint32_t size
                )=0;
            }
        }
    }
}
```

Arguments

<i>outVertexBuffer</i>	Pointer to which the generated vertex buffer returns
<i>size</i>	<i>Buffer</i> size

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This generates a vertex buffer. To delete the generated instance, use [sce::SampleUtil::destroy\(\)](#).

SCE CONFIDENTIAL

createUniformBuffer

Create uniform buffer.

Definition

```
#include <loader.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsLoader {
                virtual int createUniformBuffer(
                    UniformBuffer **outVertexBuffer,
                    uint32_t size
                )=0;
            }
        }
    }
}
```

Arguments

<i>outVertexBuffer</i>	Pointer to which the generated uniform buffer returns
<i>size</i>	Buffer size

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This generates a uniform buffer. To delete the generated instance, use [sce::SampleUtil::destroy\(\)](#).

SCE CONFIDENTIAL

createBuffer

Create buffer.

Definition

```
#include <loader.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsLoader {
                virtual int createBuffer(
                    Buffer **outDataBuffer,
                    size_t size,
                    BufferAccessMode accessMode,
                    uint32_t bufferBindFlags,
                    size_t elementSize
                )=0;
            }
        }
    }
}
```

Arguments

<i>outDataBuffer</i>	Pointer to which the generated uniform buffer returns
<i>size</i>	<i>Buffer</i> size
<i>accessMode</i>	<i>Buffer</i> access mode
<i>bufferBindFlags</i>	Bind flag (bit OR of BufferBindFlag values)
<i>elementSize</i>	Element size

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

Creates a buffer for storing one-dimensional number data. To delete the generated instance, use [sce::SampleUtil::destroy\(\)](#).

createTexture2dBuffer

Create a two-dimensional texture buffer.

Definition

```
#include <loader.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsLoader {
                virtual int createTexture2dBuffer(
                    Texture2dBuffer **outTexture2dBuffer,
                    BufferFormat format,
                    uint32_t width,
                    uint32_t height,
                    uint32_t mipCount,
                    BufferAccessMode accessMode,
                    uint32_t bufferBindFlags,
                    MultisampleMode multiSampleMode
                )=0;
            }
        }
    }
}
```

Arguments

<i>outTexture2dBuffer</i>	Pointer to which the generated two-dimensional texture buffer is returned
<i>format</i>	Buffer element format
<i>width</i>	Texture buffer width
<i>height</i>	Texture buffer height
<i>mipCount</i>	Number of mip maps in a texture buffer
<i>accessMode</i>	Buffer access mode
<i>bufferBindFlags</i>	Bind flag (bit OR of BufferBindFlag values)
<i>multiSampleMode</i>	Multisample mode of a texture buffer

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

Creates a buffer for storing a two-dimensional texture. To delete the generated instance, use [sce::SampleUtil::destroy\(\)](#).

SCE CONFIDENTIAL

createRenderTarget

Create render target.

Definition

```
#include <loader.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsLoader {
                virtual int createRenderTarget(
                    RenderTarget **outRenderTarget,
                    uint32_t width,
                    uint32_t height
                )=0;
            }
        }
    }
}
```

Arguments

<i>outRenderTarget</i>	Pointer to which the generated render target returns
<i>width</i>	Width of the render target
<i>height</i>	Height of the render target

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This generates a render target. To delete the generated instance, use [sce::SampleUtil::destroy\(\)](#).

SCE CONFIDENTIAL

createDepthStencilSurface

Create depth and stencil target.

Definition

```
#include <loader.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsLoader {
                virtual int createDepthStencilSurface(
                    DepthStencilSurface*
                    **outDepthStencilSurface,
                    uint32_t width,
                    uint32_t height
                )=0;
            }
        }
    }
}
```

Arguments

<i>outDepthStencilSurface</i>	Pointer to which the generated depth and stencil target returns
<i>width</i>	Width of a depth and stencil target
<i>height</i>	Height of a depth and stencil target

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This generates a depth and stencil target. To delete the generated instance, use [sce::SampleUtil::destroy\(\)](#).

SCE CONFIDENTIAL

createTextureFromFile

This generates a texture from an image file.

Definition

```
#include <loader.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsLoader {
                virtual int createTextureFromFile(
                    Texture **outTexture,
                    const char *filename
                )=0;
            }
        }
    }
}
```

Arguments

<i>outTexture</i>	Pointer to which the generated texture is returned
<i>filename</i>	Image file name

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

The supported formats are 24 bit uncompressed TGA (stores form left to right), 32 bit uncompressed TGA (stores from left to right), GNF, 8 bit BMP, 24 bit BMP, and 32 bit BMP. To delete the generated instance, use [sce::SampleUtil::destroy\(\)](#).

SCE CONFIDENTIAL

createTextureFromMemory

Create texture.

Definition

```
#include <loader.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsLoader {
                virtual int createTextureFromMemory(
                    Texture **outTexture,
                    const void *image,
                    uint32_t imageSize
                )=0;
            }
        }
    }
}
```

Arguments

<i>outTexture</i>	Pointer to which the generated texture is returned
<i>image</i>	Image data
<i>imageSize</i>	Size (bytes) of the image data

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This generates a texture from an image data in memory. To delete the generated instance, use [sce::SampleUtil::destroy\(\)](#).

SCE CONFIDENTIAL

createTexture

Create texture.

Definition

```
#include <loader.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class GraphicsLoader {
                virtual int createTexture(
                    Texture **outTexture,
                    TextureFormat texFormat,
                    uint32_t width,
                    uint32_t height,
                    uint32_t mipCount
                )=0;
            }
        }
    }
}
```

Arguments

<i>outTexture</i>	Pointer to which the generated texture is returned
<i>texFormat</i>	Format of a texture
<i>width</i>	Width of a texture
<i>height</i>	Height of a texture
<i>mipCount</i>	Number of mip maps in a texture

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This generates a texture. To delete the generated instance, use [sce::SampleUtil::destroy\(\)](#).

sce::SampleUtil::Graphics::Gxm::Comm onDialogUpdateParam

Summary

sce::SampleUtil::Graphics::Gxm::CommonDialogUpdateParam

Parameter for sceCommonDialogUpdate.

Definition

```
#include <platform_gxm.h>
class CommonDialogUpdateParam {};
```

Description

This is a parameter for sceCommonDialogUpdate.

Methods Summary

Methods	Description
CommonDialogUpdateParam	Constructor.
initialize	Initializes the parameter for sceCommonDialogUpdate.
finalize	Termination processing of the parameter for sceCommonDialogUpdate.

Constructors and Destructors

CommonDialogUpdateParam

Constructor.

Definition

```
#include <platform_gxm.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Gxm {
                class CommonDialogUpdateParam {
                    inline CommonDialogUpdateParam(void);
                }
            }
        }
    }
}
```

Return Values

None

Description

This is a constructor.

Public Instance Methods

initialize

Initializes the parameter for sceCommonDialogUpdate.

Definition

```
#include <platform_gxm.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Gxm {
                class CommonDialogUpdateParam {
                    int initialize(
                        RenderTarget *renderTarget,
                        DepthStencilSurface
                        *depthStencilSurface
                    );
                };
            }
        }
    }
}
```

Arguments

<i>renderTarget</i>	Pointer to a render target
<i>depthStencilSurface</i>	Pointer to a depth and stencil surface

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This initializes the parameter for sceCommonDialogUpdate.

SCE CONFIDENTIAL

finalize

Termination processing of the parameter for sceCommonDialogUpdate.

Definition

```
#include <platform_gxm.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            namespace Gxm {
                class CommonDialogUpdateParam {
                    inline int finalize(void);
                }
            }
        }
    }
}
```

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This executes termination processing of the parameter for sceCommonDialogUpdate.

sce::SampleUtil::Graphics::Parameter

000004892117

Summary

sce::SampleUtil::Graphics::Parameter

[Parameter](#) class of a shader.

Definition

```
#include <program.h>
class Parameter {};
```

Description

This is the parameter class of a shader.

Methods Summary

Methods	Description
getSemantic	Returns the semantic of an attribute parameter.
getComponentCount	Gets the number of the scalar components of each array element.
getArraySize	Gets the size of an array.
isUseDefaultBuf	Confirms the use of the default uniform buffer.
getResourceIndex	Gets a resource index.
getContainerIndex	Gets a container index.
getSemanticIndex	Returns the semantic index of attribute parameters.
setUniformDataF	Writes the value of a parameter to the appropriate offset on a uniform buffer.
getName	Returns the name of a parameter.
getCategory	Returns the category of a parameter.
isValid	Confirms whether this parameter is valid.
getType	Returns the type of a parameter.

Public Instance Methods

getSemantic

Returns the semantic of an attribute parameter.

Definition

```
#include <program.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Parameter {
                virtual ParameterSemantic getSemantic(void)=0
            const;
        }
    }
}
```

Return Values

Semantic of an attribute parameter. In the case of other than an attribute parameter, PARAMETER_SEMANTIC_NONE is returned.

Description

This returns the semantic of an attribute parameter.

SCE CONFIDENTIAL

getComponentCount

Gets the number of the scalar components of each array element.

Definition

```
#include <program.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Parameter {
                virtual uint32_t getComponentCount(void)=0 const;
            }
        }
    }
}
```

Return Values

Number of the scalar components of each array element

Description

This gets the number of the scalar components of each array element.

SCE CONFIDENTIAL

getArraySize

Gets the size of an array.

Definition

```
#include <program.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Parameter {
                virtual uint32_t getArraySize(void)=0 const;
            }
        }
    }
}
```

Return Values

Size of an array

Description

This gets the size of an array.

SCE CONFIDENTIAL

isUseDefaultBuf

Confirms the use of the default uniform buffer.

Definition

```
#include <program.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Parameter {
                virtual bool isUseDefaultBuf(void)=0 const;
            }
        }
    }
}
```

Return Values

true if used, and false if not.

Description

This confirms whether the default uniform buffer is used.

SCE CONFIDENTIAL

getResourceIndex

Gets a resource index.

Definition

```
#include <program.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Parameter {
                virtual uint32_t getResourceIndex(void)=0 const;
            }
        }
    }
}
```

Return Values

[Resource](#) index

Description

This returns a different value according to the category of a parameter. - Attribute: Register index - Uniform: Word (32-bit) offset from the head of a buffer - Sampler: [Texture](#) unit index

SCE CONFIDENTIAL

getContainerIndex

Gets a container index.

Definition

```
#include <program.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Parameter {
                virtual uint32_t getContainerIndex(void)=0 const;
            }
        }
    }
}
```

Return Values

Container index

Description

In the case of a uniform buffer, the index of uniform buffers is returned. As for other parameters, the returned values are not determined.

SCE CONFIDENTIAL

getSemanticIndex

Returns the semantic index of attribute parameters.

Definition

```
#include <program.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Parameter {
                virtual uint32_t getSemanticIndex(void)=0 const;
            }
        }
    }
}
```

Return Values

Semantic index of attribute parameters. In the case of other than an attribute parameter, 0 is returned.

Description

This returns the semantic index of attribute parameters.

SCE CONFIDENTIAL

setUniformDataF

Writes the value of a parameter to the appropriate offset on a uniform buffer.

Definition

```
#include <program.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Parameter {
                virtual int setUniformDataF(
                    void *uniformBuffer,
                    const float *sourceData
                )=0 const;
            }
        }
    }
}
```

Arguments

<i>uniformBuffer</i>	The starting address of the buffer of the write destination
<i>sourceData</i>	The data to be written

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

This writes the value of this parameter to the appropriate offset on a uniform buffer. sourceData requires [getComponentCount\(\)](#) * [getArraySize\(\)](#) elements.

SCE CONFIDENTIAL

getName

Returns the name of a parameter.

Definition

```
#include <program.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Parameter {
                virtual const char *getName(void)=0 const;
            }
        }
    }
}
```

Return Values

Name of a parameter

Description

This returns the name of a parameter.

SCE CONFIDENTIAL

getCategory

Returns the category of a parameter.

Definition

```
#include <program.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Parameter {
                virtual ParameterCategory getCategory(void)=0
            const;
        }
    }
}
```

Return Values

Category of a parameter

Description

This returns the category of a parameter.

SCE CONFIDENTIAL

isValid

Confirms whether this parameter is valid.

Definition

```
#include <program.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Parameter {
                virtual bool isValid(void)=0 const;
            }
        }
    }
}
```

Return Values

true if valid, and false if not.

Description

This confirms whether this parameter is valid.

SCE CONFIDENTIAL

getType

Returns the type of a parameter.

Definition

```
#include <program.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Parameter {
                virtual ParameterType getType(void)=0 const;
            }
        }
    }
}
```

Return Values

Format of a parameter

Description

This returns the format of a parameter.

SCE CONFIDENTIAL

sce::SampleUtil::Graphics::Program

000004892117

Summary

sce::SampleUtil::Graphics::Program

ID of a shader program.

Definition

```
#include <program.h>
class Program : public sce::SampleUtil::Resource {};
```

Description

This is the ID of a shader program.

Methods Summary

Methods	Description
isValid	Confirms whether a shader program is valid.
getParameterCount	Gets the number of parameters.
findParameterByName	Returns the parameter of the specified name.
findParameterBySemantic	Returns the parameter of the specified semantic.
getParameter	Returns the parameter of the specified index.

Public Instance Methods

isValid

Confirms whether a shader program is valid.

Definition

```
#include <program.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Program {
                virtual bool isValid(void)=0 const;
            }
        }
    }
}
```

Return Values

This returns true if valid, and false if not.

Description

This confirms whether a shader program is valid.

SCE CONFIDENTIAL

getParameterCount

Gets the number of parameters.

Definition

```
#include <program.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Program {
                virtual uint32_t getParameterCount(void)=0 const;
            }
        }
    }
}
```

Return Values

Number of parameters

Description

This gets the number of parameters.

SCE CONFIDENTIAL

findParameterByName

Returns the parameter of the specified name.

Definition

```
#include <program.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Program {
                virtual const Parameter *findParameterByName(
                    const char *name
                )=0 const;
            }
        }
    }
}
```

Arguments

name [Parameter](#) name

Return Values

[Parameter](#). If not found, an invalid parameter will be returned.

Description

This returns the parameter of the specified name.

SCE CONFIDENTIAL

findParameterBySemantic

Returns the parameter of the specified semantic.

Definition

```
#include <program.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Program {
                virtual const Parameter *findParameterBySemantic(
                    ParameterSemantic semantic,
                    uint32_t semanticIndex
                )=0 const;
            }
        }
    }
}
```

Arguments

<i>semantic</i>	Semantic of an parameter
<i>semanticIndex</i>	Semantic index of a parameter

Return Values

[Parameter](#). If not found, an invalid parameter will be returned.

Description

This returns the parameter of the specified semantic.

SCE CONFIDENTIAL

getParameter

Returns the parameter of the specified index.

Definition

```
#include <program.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Program {
                virtual const Parameter *getParameter(
                    uint32_t index
                )=0 const;
            }
        }
    }
}
```

Arguments

index Index of parameters

Return Values

[Parameter](#). Returns an invalid parameter if the index is out of range.

Description

This returns the parameter of the specified index.

**sce::SampleUtil::Graphics::SpriteRende
rer**

000004892117

Summary

sce::SampleUtil::Graphics::SpriteRenderer

2D sprite rendering class

Definition

```
#include <sprite.h>
class SpriteRenderer : public sce::SampleUtil::Resource {};
```

Description

This is a 2D sprite rendering class.

Methods Summary

Methods	Description
setRenderTargetSize	Sets the size of rendering target.
fillRect	Draws a square.
drawRect	Draws a square.
fillOval	Draws an ellipse.
drawOval	Draws an ellipse.
drawLine	Draws a line segment.
drawTexture	Draws a texture.
drawTexture	Draws a texture.
drawTextureYuy2	Draws a texture in YUY2 format.
drawTextureYuy2	Draws a texture in YUY2 format.
drawPoints	Draws a dot.
drawString	Draws a character texture.
getStringTextureSize	Gets the size of a character texture.
drawDebugString	Drawing a debug character string.
drawDebugString	Drawing a debug character string.
drawDebugStringf	Drawing a debug character string.
drawDebugStringf	Drawing a debug character string.
getWidthOfDebugChar	Gets width of a debug character.

Public Instance Methods

setRenderTargetSize

Sets the size of rendering target.

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class SpriteRenderer {
                virtual void setRenderTargetSize(
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    renderTargetSize
                )=0;
            }
        }
    }
}
```

Arguments

renderTargetSize Rendering target size

Return Values

None

Description

This sets the size of a rendering target. If drawString is to be used, make sure to execute setRenderTargetSize in advance.

SCE CONFIDENTIAL

fillRect

Draws a square.

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class SpriteRenderer {
                virtual int fillRect(
                    GraphicsContext *context,
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    position,
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    size,
                    sce::Vectormath::Simd::Aos::Vector4_arg
                    rgba = sce::Vectormath::Simd::Aos::Vector4(1.0f),
                    float depth = 0.0f
                )=0;
            }
        }
    }
}
```

Arguments

<i>context</i>	Pointer to GraphicsContext
<i>position</i>	Position of drawing (Specifies the upper left coordinate)
<i>size</i>	Size of a square (Specifies the width and height)
<i>rgba</i>	Color of a square (Specifies RGBA)
<i>depth</i>	Depth value

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid

Description

This draws a square. The square is filled with the specified color.

SCE CONFIDENTIAL

drawRect

Draws a square.

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class SpriteRenderer {
                virtual int drawRect(
                    GraphicsContext *context,
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    position,
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    size,
                    sce::Vectormath::Simd::Aos::Vector4_arg
                    rgba = sce::Vectormath::Simd::Aos::Vector4(1.0f),
                    float depth = 0.0f,
                    uint32_t width = 8
                )=0;
            }
        }
    }
}
```

Arguments

<i>context</i>	Pointer to GraphicsContext
<i>position</i>	Position of drawing (Specifies the upper left coordinate)
<i>size</i>	Size of a square (Specifies the width and height)
<i>rgba</i>	Color of a square (Specifies RGBA)
<i>depth</i>	Depth value
<i>width</i>	Line width

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid

Description

This draws a square.

fillOval

Draws an ellipse.

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class SpriteRenderer {
                virtual int fillOval(
                    GraphicsContext *context,
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    position,
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    size,
                    sce::Vectormath::Simd::Aos::Vector4_arg
                    rgba = sce::Vectormath::Simd::Aos::Vector4(1.0f),
                    float depth = 0.0f
                )=0;
            }
        }
    }
}
```

Arguments

<i>context</i>	Pointer to GraphicsContext
<i>position</i>	Position of drawing (Specifies the upper left coordinate)
<i>size</i>	Size of an ellipse (Specifies the width and height)
<i>rgba</i>	Color of an ellipse (Specifies RGBA)
<i>depth</i>	Depth value

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid

Description

This draws an ellipse. The ellipse is filled with the specified color.

SCE CONFIDENTIAL

drawOval

Draws an ellipse.

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class SpriteRenderer {
                virtual int drawOval(
                    GraphicsContext *context,
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    position,
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    size,
                    sce::Vectormath::Simd::Aos::Vector4_arg
                    rgba = sce::Vectormath::Simd::Aos::Vector4(1.0f),
                    float depth = 0.0f,
                    uint32_t width = 1
                )=0;
            }
        }
    }
}
```

Arguments

<i>context</i>	Pointer to GraphicsContext
<i>position</i>	Position of drawing (Specifies the upper left coordinate)
<i>size</i>	Size of an ellipse (Specifies the width and height)
<i>rgba</i>	Color of an ellipse (Specifies RGBA)
<i>depth</i>	Depth value
<i>width</i>	Line width

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid

Description

This draws an ellipse.

drawLine

Draws a line segment.

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class SpriteRenderer {
                virtual int drawLine(
                    GraphicsContext *context,
                    sce::Vectormath::Simd::Aos::Vector2_arg
begin,
end,
rgba = sce::Vectormath::Simd::Aos::Vector4(1.0f),
float depth = 0.0f,
uint32_t width = 1
                )=0;
            }
        }
    }
}
```

Arguments

<i>context</i>	Pointer to GraphicsContext
<i>begin</i>	Line segment beginning position
<i>end</i>	Line segment ending position
<i>rgba</i>	Color of a line segment (Specifies RGBA)
<i>depth</i>	Depth value
<i>width</i>	Line width

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid

Description

This draws a line segment.

SCE CONFIDENTIAL

drawTexture

Draws a texture.

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class SpriteRenderer {
                virtual int drawTexture(
                    GraphicsContext *context,
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    position,
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    size,
                    Texture *texture,
                    sce::Vectormath::Simd::Aos::Vector4_arg
                    colorCoeff = sce::Vectormath::Simd::Aos::Vector4(1.0f),
                    float depth = 0.0f
                )=0;
            }
        }
    }
}
```

Arguments

<i>context</i>	Pointer to GraphicsContext
<i>position</i>	Position of drawing (Specifies the upper left coordinate)
<i>size</i>	Size of the render within the render target (specify width and height)
<i>texture</i>	Pointer to Texture
<i>colorCoeff</i>	Color coefficient of a texture (Specifies RGBA)
<i>depth</i>	Depth value

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid

Description

This draws a texture.

SCE CONFIDENTIAL

drawTexture

Draws a texture.

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class SpriteRenderer {
                virtual int drawTexture(
                    GraphicsContext *context,
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    position,
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    size,
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    textureOffset,
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    sizeInTexture,
                    Texture *texture,
                    sce::Vectormath::Simd::Aos::Vector4_arg
                    colorCoeff = sce::Vectormath::Simd::Aos::Vector4(1.0f),
                    float depth = 0.0f
                )=0;
            }
        }
    }
}
```

Arguments

<i>context</i>	Pointer to GraphicsContext
<i>position</i>	Position of drawing (Specifies the upper left coordinate)
<i>size</i>	Size of the render within the render target (specify width and height)
<i>textureOffset</i>	Offset within the texture
<i>sizeInTexture</i>	Size in the texture
<i>texture</i>	Pointer to Texture
<i>colorCoeff</i>	Color coefficient of a texture (Specifies RGBA)
<i>depth</i>	Depth value

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid

Description

This draws a texture.

SCE CONFIDENTIAL

drawTextureYuy2

Draws a texture in YUY2 format.

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class SpriteRenderer {
                virtual int drawTextureYuy2(
                    GraphicsContext *context,
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    position,
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    size,
                    Texture *texture,
                    sce::Vectormath::Simd::Aos::Vector4_arg
                    colorCoeff = sce::Vectormath::Simd::Aos::Vector4(1.0f),
                    float depth = 0.0f
                )=0;
            }
        }
    }
}
```

Arguments

<i>context</i>	Pointer to GraphicsContext
<i>position</i>	Position of drawing (Specifies the upper left coordinate)
<i>size</i>	Size of the render within the render target (specify width and height)
<i>texture</i>	Pointer to Texture
<i>colorCoeff</i>	Color coefficient of a texture (Specifies RGBA)
<i>depth</i>	Depth value

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid

Description

The RGBA elements of the texture are considered YUY2 format YUYV and are converted to RGB. 1.0f will be set for A.

SCE CONFIDENTIAL

drawTextureYuy2

Draws a texture in YUY2 format.

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class SpriteRenderer {
                virtual int drawTextureYuy2(
                    GraphicsContext *context,
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    position,
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    size,
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    textureOffset,
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    sizeInTexture,
                    Texture *texture,
                    sce::Vectormath::Simd::Aos::Vector4_arg
                    colorCoeff = sce::Vectormath::Simd::Aos::Vector4(1.0f),
                    float depth = 0.0f
                )=0;
            }
        }
    }
}
```

Arguments

<i>context</i>	Pointer to GraphicsContext
<i>position</i>	Position of drawing (Specifies the upper left coordinate)
<i>size</i>	Size of the render within the render target (specify width and height)
<i>textureOffset</i>	Offset within the texture
<i>sizeInTexture</i>	Size in the texture
<i>texture</i>	Pointer to Texture
<i>colorCoeff</i>	Color coefficient of a texture (Specifies RGBA)
<i>depth</i>	Depth value

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid

Description

The RGBA elements of the texture are considered YUY2 format YUYV and are converted to RGB. 1.0f will be set for A.

SCE CONFIDENTIAL

drawPoints

Draws a dot.

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class SpriteRenderer {
                virtual int drawPoints(
                    GraphicsContext *context,
                    float psize,
                    Texture *texture,
                    VertexBuffer *vertices,
                    IndexBuffer *indices,
                    uint32_t numIndices,
                    sce::Vectormath::Simd::Aos::Vector4_arg
                    colorCoeff = sce::Vectormath::Simd::Aos::Vector4(1.0f),
                    float depth = 0.0f
                )=0;
            }
        }
    }
}
```

Arguments

<i>context</i>	Pointer to GraphicsContext
<i>psize</i>	Point size
<i>texture</i>	Pointer to Texture
<i>vertices</i>	Pointer to VertexBuffer
<i>indices</i>	Pointer to IndexBuffer
<i>numIndices</i>	Number of indices
<i>colorCoeff</i>	Color coefficient of a texture (Specifies RGBA)
<i>depth</i>	Depth value

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid

Description

This draws a dot.

SCE CONFIDENTIAL

drawString

Draws a character texture.

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class SpriteRenderer {
                virtual int drawString(
                    GraphicsContext *context,
                    Font *font,
                    const uint16_t *ucs2Charcode,
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    position,
                    sce::Vectormath::Simd::Aos::Vector4_arg
                    colorCoeff,
                    float depth = 0.0f,
                    float scale = 1.0f
                )=0;
            }
        }
    }
}
```

Arguments

<i>context</i>	Pointer to GraphicsContext
<i>font</i>	Pointer to Font
<i>ucs2Charcode</i>	Pointer to a UCS2 character code
<i>position</i>	Position of drawing (Specifies the upper left coordinate)
<i>colorCoeff</i>	Color coefficient of a texture (Specifies RGBA)
<i>depth</i>	Depth value
<i>scale</i>	Scale (enlargement factor)

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid

Description

This creates a character texture by pointer specification of a UCS2 character code and draws it. It is possible to specify multiple characters. Make sure to always set the termination of a UCS2 character code to NULL.

*When `drawString` is used, it is necessary to execute `setRenderTargetSize` after initializing [SpriteRenderer](#).

getStringTextureSize

Gets the size of a character texture.

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class SpriteRenderer {
                virtual sce::Vectormath::Simd::Aos::Vector2
getStringTextureSize(
                    Font *font,
                    const uint16_t *ucs2Charcode,
                    float scale = 1.0f
                )=0;
            }
        }
    }
}
```

Arguments

<i>font</i>	Pointer to Font
<i>ucs2Charcode</i>	Pointer to a UCS2 character code
<i>scale</i>	Scale (enlargement factor)

Return Values

Value	Description
<code>sce::Vectormath::Simd::Aos::Vector2</code>	Character texture size

Description

This returns the size of a character texture created by pointer specification of a UCS2 character code. It is possible to specify multiple characters. Make sure to always set the termination of a UCS2 character code to NULL.

SCE CONFIDENTIAL

drawDebugString

Drawing a debug character string.

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class SpriteRenderer {
                virtual int drawDebugString(
                    GraphicsContext *context,
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    position,
                    float fontHeight,
                    sce::Vectormath::Simd::Aos::Vector4_arg
                    colorCoeff,
                    sce::Vectormath::Simd::Aos::Vector4_arg
                    backgroundColor,
                    float depth,
                    const char *string
                )=0;
            }
        }
    }
}
```

Arguments

<i>context</i>	Pointer to GraphicsContext
<i>position</i>	Position of drawing (Specifies the upper left coordinate)
<i>fontHeight</i>	Height of the font drawn
<i>colorCoeff</i>	Color coefficient of a texture (Specifies RGBA)
<i>backgroundColor</i>	Background color (specifies RGBA)
<i>depth</i>	Depth value
<i>string</i>	Character string to render

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid

Description

Draws a C Language style character string (Char array terminated with NULL) on the screen.

drawDebugString

Drawing a debug character string.

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class SpriteRenderer {
                virtual int drawDebugString(
                    GraphicsContext *context,
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    position,
                    float fontHeight,
                    sce::Vectormath::Simd::Aos::Vector4_arg
                    colorCoeff,
                    const char *string
                )=0;
            }
        }
    }
}
```

Arguments

<i>context</i>	Pointer to GraphicsContext
<i>position</i>	Position of drawing (Specifies the upper left coordinate)
<i>fontHeight</i>	Height of the font drawn
<i>colorCoeff</i>	Color coefficient of a texture (Specifies RGBA)
<i>string</i>	Character string to render

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid

Description

Draws a C Language style character string (Char array terminated with NULL) on the screen.

SCE CONFIDENTIAL

drawDebugStringf

Drawing a debug character string.

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class SpriteRenderer {
                virtual int drawDebugStringf(
                    GraphicsContext *context,
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    position,
                    float fontHeight,
                    sce::Vectormath::Simd::Aos::Vector4_arg
                    colorCoeff,
                    sce::Vectormath::Simd::Aos::Vector4_arg
                    backgroundColor,
                    float depth,
                    const char *format,
                    ...
                    )=0;
            }
        }
    }
}
```

Arguments

<i>context</i>	Pointer to GraphicsContext
<i>position</i>	Position of drawing (Specifies the upper left coordinate)
<i>fontHeight</i>	Height of the font drawn
<i>colorCoeff</i>	Color coefficient of a texture (Specifies RGBA)
<i>backgroundColor</i>	Background color (specifies RGBA)
<i>depth</i>	Depth value (set 1 to match other debugString functions)
<i>format</i>	Character string designated in the C Language style (Char array terminated with NULL)
...	Variable arguments

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid

Description

This draws a character string designated in the C Language style (Char array terminated with NULL) on the screen.

drawDebugStringf

Drawing a debug character string.

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class SpriteRenderer {
                virtual int drawDebugStringf(
                    GraphicsContext *context,
                    sce::Vectormath::Simd::Aos::Vector2_arg
                    position,
                    float fontHeight,
                    sce::Vectormath::Simd::Aos::Vector4_arg
                    colorCoeff,
                    const char *format,
                    ...
                    )=0;
            }
        }
    }
}
```

Arguments

<i>context</i>	Pointer to GraphicsContext
<i>position</i>	Position of drawing (Specifies the upper left coordinate)
<i>fontHeight</i>	Height of the font drawn
<i>colorCoeff</i>	Color coefficient of a texture (Specifies RGBA)
<i>format</i>	Character string designated in the C Language style (Char array terminated with NULL)
...	Variable arguments

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid

Description

This draws a character string designated in the C Language style (Char array terminated with NULL) on the screen.

getWidthOfDebugChar

Gets width of a debug character.

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class SpriteRenderer {
                virtual float getWidthOfDebugChar(
                    float charHeight
                )=0;
            }
        }
    }
}
```

Arguments

charHeight Height of a debug character

Return Values

Value	Description
float	Width of a debug character

Description

Returns character width for the designated height of specified characters in the current render target.

sce::SampleUtil::Graphics::Object3dRe nderer

000004892117

Summary

sce::SampleUtil::Graphics::Object3dRenderer

3D object renderer class

Definition

```
#include <sprite.h>
class Object3dRenderer : public sce::SampleUtil::Resource {};
```

Description

This is a class for rendering simple 3D objects.

Methods Summary

Methods	Description
fillCube	Cube render.
drawCube	Cube render.
drawLine	Draws a line segment.
drawSphere	Sphere rendering.
fillSphere	Sphere rendering.
fillCylinder	Cylinder rendering.
drawCylinder	Cylinder rendering.

Public Instance Methods

fillCube

Cube render.

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Object3dRenderer {
                virtual int fillCube(
                    GraphicsContext *context,
                    sce::Vectormath::Simd::Aos::Matrix4_arg
world,
                    sce::Vectormath::Simd::Aos::Matrix4_arg
view,
                    sce::Vectormath::Simd::Aos::Matrix4_arg
projection,
                    sce::Vectormath::Simd::Aos::Vector3
lightPosition,
                    sce::Vectormath::Simd::Aos::Vector4
color,
                    float ambient,
                    float shininess
                )=0;
            }
        }
    }
}
```

Arguments

<i>context</i>	Pointer to GraphicsContext
<i>world</i>	Transformation matrix for world coordinate system
<i>view</i>	View matrix
<i>projection</i>	Projection matrix
<i>lightPosition</i>	Point light position
<i>color</i>	Color
<i>ambient</i>	Ambient value for phone shader
<i>shininess</i>	Shininess of phone shader

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

Renders a filled cube in a 3D space. The original cube will have the center as the starting point, the length of each side will be 1, and all sides will be parallel to the coordinate axes.

SCE CONFIDENTIAL

drawCube

Cube render.

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Object3dRenderer {
                virtual int drawCube(
                    GraphicsContext *context,
                    sce::Vectormath::Simd::Aos::Matrix4_arg
world,
                    sce::Vectormath::Simd::Aos::Matrix4_arg
view,
                    sce::Vectormath::Simd::Aos::Matrix4_arg
projection,
                    sce::Vectormath::Simd::Aos::Vector4
color,
                    uint32_t lineWidth
                )=0;
            }
        }
    }
}
```

Arguments

<i>context</i>	Pointer to GraphicsContext
<i>world</i>	Transformation matrix for world coordinate system
<i>view</i>	View matrix
<i>projection</i>	Projection matrix
<i>color</i>	Color
<i>lineWidth</i>	Line width

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

Renders a wire frame cube in a 3D space. The original cube will have the center as the starting point, the length of each side will be 1, and all sides will be parallel to the coordinate axes.

SCE CONFIDENTIAL

drawLine

Draws a line segment.

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Object3dRenderer {
                virtual int drawLine(
                    GraphicsContext *context,
                    sce::Vectormath::Simd::Aos::Matrix4_arg
                    view,
                    sce::Vectormath::Simd::Aos::Matrix4_arg
                    projection,
                    sce::Vectormath::Simd::Aos::Vector3
                    begin,
                    sce::Vectormath::Simd::Aos::Vector3 end,
                    sce::Vectormath::Simd::Aos::Vector4
                    color,
                    uint32_t lineWidth
                )=0;
            }
        }
    }
}
```

Arguments

<i>context</i>	Pointer to GraphicsContext
<i>view</i>	View matrix
<i>projection</i>	Projection matrix
<i>begin</i>	Line segment beginning position
<i>end</i>	Line segment ending position
<i>color</i>	Color
<i>lineWidth</i>	Line width

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

Renders a line segment in 3D space.

SCE CONFIDENTIAL

drawSphere

Sphere rendering.

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Object3dRenderer {
                virtual int drawSphere(
                    GraphicsContext *context,
                    sce::Vectormath::Simd::Aos::Matrix4_arg
world,
                    sce::Vectormath::Simd::Aos::Matrix4_arg
view,
                    sce::Vectormath::Simd::Aos::Matrix4_arg
projection,
                    sce::Vectormath::Simd::Aos::Vector4
color,
                    uint32_t lineWidth
                )=0;
            };
        }
    }
}
```

Arguments

<i>context</i>	Pointer to GraphicsContext
<i>world</i>	Transformation matrix for world coordinate system
<i>view</i>	View matrix
<i>projection</i>	Projection matrix
<i>color</i>	Color
<i>lineWidth</i>	Line width

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

Renders a wire frame sphere in 3D space. The original sphere has its center at the origin with a diameter of 1.

SCE CONFIDENTIAL

fillSphere

Sphere rendering.

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Object3dRenderer {
                virtual int fillSphere(
                    GraphicsContext *context,
                    sce::Vectormath::Simd::Aos::Matrix4_arg
world,
                    sce::Vectormath::Simd::Aos::Matrix4_arg
view,
                    sce::Vectormath::Simd::Aos::Matrix4_arg
projection,
                    sce::Vectormath::Simd::Aos::Vector3
lightPosition,
                    sce::Vectormath::Simd::Aos::Vector4
color,
                    float ambient,
                    float shininess
                )=0;
            }
        }
    }
}
```

Arguments

<i>context</i>	Pointer to GraphicsContext
<i>world</i>	Transformation matrix for world coordinate system
<i>view</i>	View matrix
<i>projection</i>	Projection matrix
<i>lightPosition</i>	Point light position
<i>color</i>	Color
<i>ambient</i>	Ambient value for phone shader
<i>shininess</i>	Shininess of phone shader

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

Renders a colored-in sphere in 3D space. The original sphere has its center at the origin with a radius of 0.5.

SCE CONFIDENTIAL

fillCylinder

Cylinder rendering.

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Object3dRenderer {
                virtual int fillCylinder(
                    GraphicsContext *context,
                    sce::Vectormath::Simd::Aos::Matrix4_arg
world,
                    sce::Vectormath::Simd::Aos::Matrix4_arg
view,
                    sce::Vectormath::Simd::Aos::Matrix4_arg
projection,
                    sce::Vectormath::Simd::Aos::Vector3
lightPosition,
                    sce::Vectormath::Simd::Aos::Vector4
color,
                    float ambient,
                    float shininess
                )=0;
            }
        }
    }
}
```

Arguments

<i>context</i>	Pointer to GraphicsContext
<i>world</i>	Transformation matrix for world coordinate system
<i>view</i>	View matrix
<i>projection</i>	Projection matrix
<i>lightPosition</i>	Point light position
<i>color</i>	Color
<i>ambient</i>	Ambient value for phone shader
<i>shininess</i>	Shininess of phone shader

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

Renders a colored-in cylinder in 3D space. The original cylinder has its circle center at the origin with a radius of 0.5, and a height of 1.0.

SCE CONFIDENTIAL

drawCylinder

Cylinder rendering.

Definition

```
#include <sprite.h>
namespace sce {
    namespace SampleUtil {
        namespace Graphics {
            class Object3dRenderer {
                virtual int drawCylinder(
                    GraphicsContext *context,
                    sce::Vectormath::Simd::Aos::Matrix4_arg world,
                    sce::Vectormath::Simd::Aos::Matrix4_arg view,
                    sce::Vectormath::Simd::Aos::Matrix4_arg projection,
                    sce::Vectormath::Simd::Aos::Vector4 color,
                    uint32_t lineWidth
                )=0;
            }
        }
    }
}
```

Arguments

<i>context</i>	Pointer to GraphicsContext
<i>world</i>	Transformation matrix for world coordinate system
<i>view</i>	View matrix
<i>projection</i>	Projection matrix
<i>color</i>	Color
<i>lineWidth</i>	Line width

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Error

Description

Renders a wire frame cylinder in 3D space. The original cylinder has its circle center at the origin with a radius of 0.5, and a height of 1.0.

SCE CONFIDENTIAL

sce::SampleUtil::Input::PadContextOpti on

Summary

sce::SampleUtil::Input::PadContextOption

Structure for initializing [PadContext](#).

Definition

```
#include <input.h>
struct PadContextOption {};
```

Description

This is the structure for initializing [PadContext](#). This is used by specifying it to the argument "option" of PadContext::initialize().

Fields

Public Instance Fields

uint32_t numBufs

Number of the buffers of the ScePadData structure retained within [PadContext](#).

Specify a value from 1 to 64. The default value is 16.

Methods Summary

Methods	Description
PadContextOption	Constructor.

Constructors and Destructors

PadContextOption

Constructor.

Definition

```
#include <input.h>
inline PadContextOption(void);
```

Return Values

None

Description

This is a constructor.

SCE CONFIDENTIAL

sce::SampleUtil::Input::TouchPadData

000004892117

Summary

sce::SampleUtil::Input::TouchPadData

Touch point data.

Definition

```
#include <input.h>
struct TouchPadData {};
```

Description

This structure stores the information for a single touch point from the touch data.

Fields

Public Instance Fields

float <i>x</i>	x relative position (0 to 1) of touch point
float <i>y</i>	y relative position (0 to 1) of touch point
uint8_t <i>id</i>	Touch report ID (0 to 127)

SCE CONFIDENTIAL

sce::SampleUtil::Input::MagnetometerD

ata

000004892117

Summary

sce::SampleUtil::Input::MagnetometerData

Magnetometer data.

Definition

```
#include <input.h>
struct MagnetometerData {};
```

Description

This structure stores magenetometer data.

Fields

Public Instance Fields

sce::Vectormath::Simd::Aos::Matrix4
northEastDownOrientation
uint8_t *magneticFieldStability*

4x4 rotation matrix for orientation based on NED (North East Down)
Value that indicates magnetic field stability.

sce::SampleUtil::Input::MotionData

000004892117

Summary

sce::SampleUtil::Input::MotionData

Motion data.

Definition

```
#include <input.h>
struct MotionData { };
```

Description

This structure stores motion data (acceleration, angular velocity, orientation).

Fields

Public Instance Fields

sce::Vectormath::Simd::Aos::Quat <i>orientation</i>	Controller orientation value.
sce::Vectormath::Simd::Aos::Vector3 <i>acceleration</i>	Acceleration sensor XYZ reading (G)
sce::Vectormath::Simd::Aos::Vector3 <i>angularVelocity</i>	Angular velocity sensor XYZ reading (rad/s)
sce::Vectormath::Simd::Aos::Vector3 <i>userInterfaceOrientation</i>	Basic device orientation based on gravity<x, y, z>
MagnetometerData <i>magnetometer</i>	Magnetometer data. Data is only stored for PlayStation®Vita.

SCE CONFIDENTIAL

sce::SampleUtil::Input::PadData

000004892117

Summary

sce::SampleUtil::Input::PadData

Output data received from the controller.

Definition

```
#include <input.h>
struct PadData { };
```

Description

This structure is for obtaining the controller data information.

Fields

Public Instance Fields

uint64_t <i>timeStamp</i>	System timestamp when the data is obtained.
bool <i>connected</i>	Controller connection flag value.
uint32_t <i>buttons</i>	Digital button data.
uint8_t <i>lx</i>	x coordinate for the left analog stick data (0 to 255)
uint8_t <i>ly</i>	y coordinate for the left analog stick data (0 to 255)
uint8_t <i>rx</i>	x coordinate for the right analog stick data (0 to 255)
uint8_t <i>ry</i>	y coordinate for the right analog stick data (0 to 255)
uint8_t <i>l2</i>	L2 analog value.
uint8_t <i>r2</i>	R2 analog value.
uint8_t <i>touchNumber</i>	Number of touch points.
TouchPadData <i>touchPadData</i>	Touch point data array.
MotionData <i>motionData</i>	Motion data.
bool <i>intercepted</i>	Flag to indicate whether or not the controller is taken by the system software.

SCE CONFIDENTIAL

sce::SampleUtil::Input::PadContext

000004892117

Summary

sce::SampleUtil::Input::PadContext

Class for handling controller operation.

Definition

```
#include <input.h>
class PadContext : public sce::SampleUtil::Resource {};
```

Description

[PadContext](#) is a class that enables applications to easily support controller operation. It is possible to perform management of controller button states/analog stick values, management of analog stick dead zones, button "Press" and "Release" state detection, motion sensor enabled/disabled settings, tilt correction enabled/disabled settings, controller orientation value resetting, angular velocity sensor deadband filter enabled/disabled settings, and controller vibration control.

Methods Summary

Methods	Description
~PadContext	Destructor.
update	Update.
isButtonDown	Function to determine whether a button is being pressed.
isButtonUp	Function to determine that a button is not currently being pressed.
isButtonPressed	Function to determine that a button has been pressed.
isButtonReleased	Function to determine that a button has been released.
getLeftStick	Function to get left analog stick data.
getRightStick	Function to get right analog stick data.
setRightAnalogStickDeadzone	Function to set right stick dead zones.
setLeftAnalogStickDeadZone	Function to set left stick dead zones.
getData	Function to obtain controller data newly held by PadContext in the last PadContext::update() call.
getPreviousUpdateData	Function to obtain controller data newly held by PadContext in the PadContext::update() call preceding the last PadContext::update() call.
enableMotionSensor	Enables/disables motion sensor.
enableMagnetometer	Enables/disables magnetometer.
enableTiltCorrection	Enables or disables tilt correction.
enableAngularVelocityDeadband	Enables/disables the deadband filter of the angular velocity sensor.
resetOrientation	Enables/disables motion sensor.
getDefaultLeftAnalogStickDeadZone	Gets the default dead zones of the left stick.
getDefaultRightAnalogStickDeadZone	Gets the default dead zones of the right stick.

Constructors and Destructors

~PadContext

Destructor.

Definition

```
#include <input.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class PadContext {
                virtual inline ~PadContext(void);
            };
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Public Instance Methods

update

Update.

Definition

```
#include <input.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class PadContext {
                virtual int update(void)=0;
            }
        }
    }
}
```

Return Values

Value	Description
<code>>=SCE_OK</code>	Success (SCE_OK when there is no controller)
<code>(<0)</code>	Error code

Description

This is the update function of [PadContext](#).

isButtonDown

Function to determine whether a button is being pressed.

Definition

```
#include <input.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class PadContext {
                virtual bool isButtonDown(
                    uint32_t buttons,
                    ButtonEventPattern pattern =
kButtonEventPatternAll
                )=0 const;
            }
        }
    }
}
```

Arguments

buttons
pattern

The enum value of Button used to judge the function
The matching pattern of the button. PATTERN_ALL is specified as the default parameter.

Return Values

Value	Description
true	The button specified in buttons is being pressed in reference to the pattern specified by pattern
false	The button specified in buttons is not being pressed in reference to the pattern specified by pattern

Description

This function refers to the matching pattern specified in the pattern argument to determine whether the button specified in the buttons argument is currently being pressed.

isButtonUp

Function to determine that a button is not currently being pressed.

Definition

```
#include <input.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class PadContext {
                virtual bool isButtonUp(
                    uint32_t buttons,
                    ButtonEventPattern pattern =
kButtonEventPatternAll
                )=0 const;
            }
        }
    }
}
```

Arguments

buttons
pattern

The enum value of Button used to judge the function
The matching pattern of the button. PATTERN_ALL is specified as the default parameter.

Return Values

Value	Description
true	The button specified in buttons is not being pressed in reference to the pattern specified by pattern
false	The button specified in buttons is being pressed in reference to the pattern specified by pattern

Description

This function refers to the matching pattern specified in the pattern argument to determine whether the button specified in the buttons argument is currently not being pressed.

isButtonPressed

Function to determine that a button has been pressed.

Definition

```
#include <input.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class PadContext {
                virtual bool isButtonPressed(
                    uint32_t buttons,
                    ButtonEventPattern pattern =
kButtonEventPatternAll
                )=0 const;
            }
        }
    }
}
```

Arguments

buttons
pattern

The enum value of Button used to judge the function
The matching pattern of the button. PATTERN_ALL is specified as the default parameter.

Return Values

Value	Description
true	The button specified in buttons has been pressed in reference to the pattern specified by pattern
false	The button specified in buttons has not been pressed in reference to the pattern specified by pattern

Description

This function refers to the matching pattern specified in the pattern argument for the button specified in the buttons arguments and compares the controller values obtained with the last [update \(\)](#) call and the [update \(\)](#) call preceding it to determine whether the button has been pressed.

isButtonReleased

Function to determine that a button has been released.

Definition

```
#include <input.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class PadContext {
                virtual bool isButtonReleased(
                    uint32_t buttons,
                    ButtonEventPattern pattern =
kButtonEventPatternAll
                )=0 const;
            }
        }
    }
}
```

Arguments

buttons
pattern

The enum value of Button used to judge the function
The matching pattern of the button. PATTERN_ALL is specified as the default parameter.

Return Values

Value	Description
true	The button specified in buttons has been released in reference to the pattern specified by pattern
false	The button specified in buttons has not been released in reference to the pattern specified by pattern

Description

This function refers to the matching pattern specified in the pattern argument for the button specified in the buttons arguments and compares the controller values obtained with the last [update \(\)](#) call and the [update \(\)](#) call preceding it to determine whether the button has been released.

SCE CONFIDENTIAL

getLeftStick

Function to get left analog stick data.

Definition

```
#include <input.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class PadContext {
                virtual const
sce::Vectormath::Simd::Aos::Vector2 &getLeftStick()=0 const;
            }
        }
    }
}
```

Return Values

Returns the value of the left analog stick in the Vector2 type. In the X and Y values of Vector2, the X direction and Y direction of the left analog stick are stored in a float value, respectively.

Description

This is the function to get left analog stick data. In the X and Y values of Vector2 of the return value, the X direction and Y direction of the left analog stick are stored in a float value, respectively. In this function, the dead zone is also taken into account; this dead zone can be changed with [setLeftAnalogStickDeadZone\(\)](#).

getRightStick

Function to get right analog stick data.

Definition

```
#include <input.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class PadContext {
                virtual const
sce::Vectormath::Simd::Aos::Vector2 &getRightStick()=0 const;
            }
        }
    }
}
```

Return Values

Returns the value of the right analog stick in the Vector2 type. In the X and Y values of Vector2, the X direction and Y direction of the right analog stick are stored in a float value, respectively.

Description

This is the function to get right analog stick data. In the X and Y values of Vector2 of the return value, the X direction and Y direction of the right analog stick are stored in a float value, respectively. In this function, the dead zone is also taken into account; this dead zone can be changed with [setRightAnalogStickDeadZone\(\)](#).

SCE CONFIDENTIAL

setRightAnalogStickDeadZone

Function to set right stick dead zones.

Definition

```
#include <input.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class PadContext {
                virtual void setRightAnalogStickDeadZone(
                    float rightAnalogStickDeadZone
                )=0;
            }
        }
    }
}
```

Arguments

rightAnalogStickDeadZone Dead zone value of the right stick

Return Values

None

Description

This is the function to set the dead zones of the right stick. The default value is set to 0.25 for PlayStation®Vita and to the value obtained from the controller for PlayStation®4. For details, see "Pad Library Overview".

SCE CONFIDENTIAL

setLeftAnalogStickDeadZone

Function to set left stick dead zones.

Definition

```
#include <input.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class PadContext {
                virtual void setLeftAnalogStickDeadZone(
                    float leftAnalogStickDeadZone
                )=0;
            }
        }
    }
}
```

Arguments

leftAnalogStickDeadZone Dead zone value of the left stick

Return Values

None

Description

This is the function to set the dead zones of the left stick. The default value is set to 0.25 for PlayStation®Vita and to the value obtained from the controller for PlayStation®4. For details, see "Pad Library Overview".

getData

Function to obtain controller data newly held by [PadContext](#) in the last [PadContext::update\(\)](#) call.

Definition

```
#include <input.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class PadContext {
                virtual int getData(
                    PadData *data,
                    uint32_t length
                )=0 const;
            }
        }
    }
}
```

Arguments

<i>data</i>	The array of the write destination of the PadData structure
<i>length</i>	Number of elements in the data array (0 to 64)

Return Values

Returns the number of controller data stored in data for success (0 if none is stored). Returns an error code for failures.

Description

This function obtains controller data newly held by [PadContext](#) in the last [PadContext::update\(\)](#) call. Obtained controller data is stored in the data array in order from newest to oldest with the value specified in length as the upper limit.

When the return value is 0, this indicates that there is no controller data newly held by [PadContext](#) for the [PadContext::update\(\)](#) call.

getPreviousUpdateData

Function to obtain controller data newly held by [PadContext](#) in the [PadContext::update\(\)](#) call preceding the last [PadContext::update\(\)](#) call.

Definition

```
#include <input.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class PadContext {
                virtual int getPreviousUpdateData(
                    PadData *data,
                    uint32_t length
                )=0 const;
            }
        }
    }
}
```

Arguments

data

[PadData](#) structure write destination array. This function stores controller data obtained with the [PadContext::update\(\)](#) call preceding the last [PadContext::update\(\)](#) call in the data array in order from newest to oldest.

length

Length of the data array

Return Values

Returns the number of padData written in the [PadContext::update\(\)](#) call preceding the last [PadContext::update\(\)](#) call for success. When length is larger than the number of controller data newly held by [PadContext](#), returns the number of controller data newly held by [PadContext](#).

Description

This function obtains controller data newly held by [PadContext](#) in the [PadContext::update\(\)](#) call preceding the last [PadContext::update\(\)](#) call. Controller data obtained with the [PadContext::update\(\)](#) call preceding the last [PadContext::update\(\)](#) call is stored in the data array in order from newest to oldest. When 1 is specified to length, only the latest controller data obtained with the [PadContext::update\(\)](#) call preceding the last [PadContext::update\(\)](#) call will be stored in data. When value specified to length is larger than the number of controller data newly held by [PadContext](#), this function stores all the controller data newly held by [PadContext](#) with the [PadContext::update\(\)](#) call preceding the last [PadContext::update\(\)](#) call. When the value specified to length is less than the number of controller data newly held by [PadContext](#), this function stores length-number of controller data.

SCE CONFIDENTIAL

enableMotionSensor

Enables/disables motion sensor.

Definition

```
#include <input.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class PadContext {
                virtual int enableMotionSensor(
                    bool bEnable
                )=0;
            }
        }
    }
}
```

Arguments

bEnable Enables or disables. Enabled if it is true. The default is true

Return Values

Returns SCE_OK for success. Returns an error code for failures.

Description

Enables/disables motion sensor. Valid values will be stored in the orientation, acceleration, and angularVelocity members of [MotionData](#) of [PadData](#).

SCE CONFIDENTIAL

enableMagnetometer

Enables/disables magnetometer.

Definition

```
#include <input.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class PadContext {
                virtual int enableMagnetometer(
                    bool bEnable
                )=0;
            }
        }
    }
}
```

Arguments

bEnable Enables or disables. Enabled if it is true. The default is true

Return Values

Returns SCE_OK for success. Returns an error code for failures.

Description

This enables/disables the magnetometer. Valid values will be stored in the magnetometer member of [MotionData](#) of [PadData](#). An error will return when the motion sensor is disabled.

SCE CONFIDENTIAL

enableTiltCorrection

Enables or disables tilt correction.

Definition

```
#include <input.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class PadContext {
                virtual int enableTiltCorrection(
                    bool bEnable
                )=0;
            }
        }
    }
}
```

Arguments

bEnable Enables or disables. Enabled if it is true. The default is false

Return Values

Returns SCE_OK for success. Returns an error code for failures.

Description

Enables/disables the feature for correcting the controller orientation data based on the acceleration sensor.

When tilt correction is enabled, tilt correction will apply to the orientation data to be stored in the orientation member of [MotionData](#) of [PadData](#).

SCE CONFIDENTIAL

enableAngularVelocityDeadband

Enables/disables the deadband filter of the angular velocity sensor.

Definition

```
#include <input.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class PadContext {
                virtual int enableAngularVelocityDeadband(
                    bool bEnable
                )=0;
            }
        }
    }
}
```

Arguments

bEnable Enables or disables. Enabled if it is true. The default is false

Return Values

Returns SCE_OK for success. Returns an error code for failures.

Description

Enables/disables the deadband filter of the angular velocity sensor (gyro sensor).

When the deadband filter is enabled, the orientation and angularVelocity members of [MotionData](#) of [PadData](#) will be calculated based on the angular velocity value after noise removal with the deadband filter.

SCE CONFIDENTIAL

resetOrientation

Enables/disables motion sensor.

Definition

```
#include <input.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class PadContext {
                virtual int resetOrientation()=0;
            }
        }
    }
}
```

Return Values

Returns SCE_OK for success. Returns an error code for failures.

Description

This function obtains the controller data array currently held by [PadContext](#). This is the buffer count designated at the time of initialization for maximum array length.

SCE CONFIDENTIAL

getDefaultLeftAnalogStickDeadZone

Gets the default dead zones of the left stick.

Definition

```
#include <input.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class PadContext {
                virtual float
getDefaultLeftAnalogStickDeadZone(void)=0;
            }
        }
    }
}
```

Return Values

Returns SCE_OK for success. Returns an error code for failures.

Description

This gets default dead zones of the left stick.

SCE CONFIDENTIAL

getDefaultRightAnalogStickDeadZone

Gets the default dead zones of the right stick.

Definition

```
#include <input.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class PadContext {
                virtual float
getDefaultValueRightAnalogStickDeadZone(void)=0;
            }
        }
    }
}
```

Return Values

Returns SCE_OK for success. Returns an error code for failures.

Description

This gets default dead zones of the right stick.

SCE CONFIDENTIAL

sce::SampleUtil::Input

000004892117

Summary

sce::SampleUtil::Input

Input-associated definitions.

Definition

```
namespace Input {}
```

Description

[sce::SampleUtil::Input](#) is the name space associated with the input of the [SampleUtil](#) library.

Variables

Public Variables

static const uint32_t	Number of controller ports.
<i>kControllerPortNum</i>	Number of controller ports

Function Summary

Function	Description
createPadContext	PadContext generation.
createControllerContext	Initialization function of ControllerContext .
createMotionContext	Create MotionContext .
createTouchContext	Generates TouchContext .

Inner Classes, Structures, and Namespaces

Item	Description
sce::SampleUtil::Input::PadContextOption	Structure for initializing PadContext .
sce::SampleUtil::Input::TouchPadData	Touch point data.
sce::SampleUtil::Input::MagnetometerData	Magnetometer data.
sce::SampleUtil::Input::MotionData	Motion data.
sce::SampleUtil::Input::PadData	Output data received from the controller.
sce::SampleUtil::Input::PadContext	Class for handling controller operation.
sce::SampleUtil::Input::ControllerContextOption	Structure for initializing ControllerContext .
sce::SampleUtil::Input::ControllerContext	Class for handling controller operation.
sce::SampleUtil::Input::MotionContextData	Structure that handles device states and motion sensor values.
sce::SampleUtil::Input::MotionContext	Class that handles motion sensor data.
sce::SampleUtil::Input::TouchContextOption	Structure for setting TouchContext .
sce::SampleUtil::Input::TouchPrimitiveGestureEvent	Structure for getting the primitive gesture event.
sce::SampleUtil::Input::TouchGestureEvent	Structure for getting the gesture event.
sce::SampleUtil::Input::TouchContext	Class to handle input data from the touch panel.

Enumerated Types

ButtonEventPattern

The enumeration type to be used for specifying a matching pattern of buttons.

Definition

```
#include <controller.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            enum ButtonEventPattern {
                kButtonEventPatternAny = 0,
                kButtonEventPatternAll
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kButtonEventPatternAny	0	If any one of the specified buttons matches, the function will return true as the return value.
kButtonEventPatternAll	N/A	If all of the specified buttons matches, the function will return true as the return value.

Description

Used for specifying for the 2nd argument "pattern" of the [PadContext](#) member functions isButtonDown(), isButtonUp(), isButtonPressed(), and isButtonReleased().

SCE CONFIDENTIAL

Button

The enum value of a button.

Definition

```
#include <controller.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            enum Button {
                kButtonSelect = (1<<0),
                kButtonStart = (1<<3),
                kButtonUp = (1<<4),
                kButtonRight = (1<<5),
                kButtonDown = (1<<6),
                kButtonLeft = (1<<7),
                kButtonL = (1<<8),
                kButtonR = (1<<9),
                kButtonTriangle = (1<<12),
                kButtonCircle = (1<<13),
                kButtonCross = (1<<14),
                kButtonSquare = (1<<15)
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kButtonSelect	(1<<0)	SELECT button
kButtonStart	(1<<3)	START button.
kButtonUp	(1<<4)	Up button.
kButtonRight	(1<<5)	Right button.
kButtonDown	(1<<6)	Down button.
kButtonLeft	(1<<7)	Left button.
kButtonL	(1<<8)	L button.
kButtonR	(1<<9)	R button.
kButtonTriangle	(1<<12)	Triangle button.
kButtonCircle	(1<<13)	Circle button.
kButtonCross	(1<<14)	Cross button.
kButtonSquare	(1<<15)	Square button.

Description

Used for specifying for the 1st argument "buttons" of the [PadContext](#) member functions isButtonDown(), isButtonUp(), isButtonPressed(), and isButtonReleased().

SCE CONFIDENTIAL

MotionContextFunctionFlag

enum value of the flag for the feature to use with the motion context

Definition

```
#include <motion.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            enum MotionContextFunctionFlag {
                kMotionContextFunctionTypeAccelerometerAndGyro =
0x00000001,
                kMotionContextFunctionTypeMagnetometer =
0x00000002
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kMotionContextFunctionTypeAccelerometerAndGyro	0x00000001	Accelerometer sensor and gyro sensor flag value.
kMotionContextFunctionTypeMagnetometer	0x00000002	Magnetometer flag value.

Description

This is used to specify the flag for the feature to use with the motion context

SCE CONFIDENTIAL

TouchPort

The touch port enum value.

Definition

```
#include <touch.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            enum TouchPort {
                kTouchPortFront = 0,
                kTouchPortBack = 1
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kTouchPortFront	0	Front touch panel.
kTouchPortBack	1	Back touch panel.

Description

This is used for the designation of the touch panel port.

SCE CONFIDENTIAL

TouchContextFunctionFlag

Function flag enum value.

Definition

```
#include <touch.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            enum TouchContextFunctionFlag {
                kTouchContextFunctionFlagTapGestureEvent =
0x00000001,
                kTouchContextFunctionFlagDragGestureEvent =
0x00000002,
                kTouchContextFunctionFlagTapAndHoldGestureEvent =
0x00000004,
                kTouchContextFunctionFlagPinchOutInGestureEvent =
0x00000008,
                kTouchContextFunctionFlagRotationGestureEvent =
0x00000010,
                kTouchContextFunctionFlagPrimitiveGestureEvent =
0x40000000,
                kTouchContextFunctionFlagRawData = 0x80000000
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kTouchContextFunctionFlagTapGestureEvent	0x00000001	Tap gesture event.
kTouchContextFunctionFlagDragGestureEvent	0x00000002	Drag gesture event.
kTouchContextFunctionFlagTapAndHoldGestureEvent	0x00000004	Tap and hold gesture event.
kTouchContextFunctionFlagPinchOutInGestureEvent	0x00000008	Pinch out-in gesture event.
kTouchContextFunctionFlagRotationGestureEvent	0x00000010	Rotation gesture event.
kTouchContextFunctionFlagPrimitiveGestureEvent	0x40000000	Primitive gesture event.
kTouchContextFunctionFlagRawData	0x80000000	Touch data.

Description

This is used for the designation of the function flag.

Functions

createPadContext

[PadContext](#) generation.

Definition

```
#include <input.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            int createPadContext(
                PadContext **outPadContext,
                const sce::SampleUtil::System::UserId userId,
                const int32_t type,
                const int32_t index,
                const PadContextOption *option = NULL
            );
        }
    }
}
```

Arguments

<i>outPadContext</i>	Pointer to be returned by the generated PadContext .
<i>userId</i>	User ID
<i>type</i>	Controller type (specify 0)
<i>index</i>	Device index (specify 0)
<i>option</i>	PadContextOption structure. This is initialized by the default value if NULL is specified.

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This generates [PadContext](#). To destroy a generated [PadContext](#), use [sce::SampleUtil::destroy\(\)](#).

SCE CONFIDENTIAL

createControllerContext

Initialization function of [ControllerContext](#).

Definition

```
#include <controller.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            int createControllerContext(
                ControllerContext **outControllerContext,
                const ControllerContextOption *option = NULL
            );
        }
    }
}
```

Arguments

<i>outControllerContext</i> <i>option</i>	Pointer to which the generated ControllerContext returns The ControllerContextOption structure. This is initialized by the default value if NULL is specified.
--	--

Return Values

Value	Description
SCE_OK	Success

Description

This is the initialization function of [ControllerContext](#). Call this first when using [ControllerContext](#).

SCE CONFIDENTIAL

createMotionContext

Create [MotionContext](#).

Definition

```
#include <motion.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            int createMotionContext(
                MotionContext **outMotionContext
            );
        }
    }
}
```

Arguments

outMotionContext Pointer to which the generated [MotionContext](#) returns

Return Values

Value	Description
(≥ 0)	Success
(<0)	Failure

Description

This executes the initialization processing of the [MotionContext](#) class.

SCE CONFIDENTIAL

createTouchContext

Generates [TouchEvent](#).

Definition

```
#include <touch.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            int createTouchContext(
                TouchContext **outTouchContext,
                const TouchContextOption *option = NULL
            );
        }
    }
}
```

Arguments

<i>outTouchContext</i>	Pointer to which the generated TouchEvent returns
<i>option</i>	Pointer to TouchContextOption . This is initialized by the default value if NULL is specified.

Return Values

Value	Description
SCE_OK	Success
<SCE_OK	Failure

Description

This executes the initialization processing of the [TouchEvent](#) class. If the argument option is omitted, all processing will be executed with the initial set values.

sce::SampleUtil::Input::ControllerContentOption

Summary

sce::SampleUtil::Input::ControllerContextOption

Structure for initializing [ControllerContext](#).

Definition

```
#include <controller.h>
struct ControllerContextOption {};
```

Description

This is the structure for initializing [ControllerContext](#). This is used by specifying it to the argument option of ControllerContext::initialize().

Fields

Public Instance Fields

uint32_t numBufs

Number of the buffers of the SceCtrlData structure retained within [ControllerContext](#).
Specify a value from 1 to 64. The default value is 64.

SCE CONFIDENTIAL

sce::SampleUtil::Input::ControllerConte

xt

000004892117

Summary

sce::SampleUtil::Input::ControllerContext

Class for handling controller operation.

Definition

```
#include <controller.h>
class ControllerContext : public sce::SampleUtil::Resource {};
```

Description

[ControllerContext](#) is a class that enable applications to easily support controller operation. The following features are provided.

- Management of the button state of the controller and analog stick value
- Management of the dead zones of the analog stick
- Indication of the "Press" and "Release" events of a button

Methods Summary

Methods	Description
update	Update ControllerContext .
isButtonDown	Function to determine whether a button is being pressed.
isButtonUp	Function to determine that a button is not currently being pressed.
isButtonPressed	Function to determine that a button has been pressed.
isButtonReleased	Function to determine that a button has been released.
getLeftStick	Function to get left analog stick data.
getRightStick	Function to get right analog stick data.
setDeadZone	Function to set analog stick dead zones.
getData	Function to obtain the controller data currently held by ControllerContext .

Public Instance Methods

update

Update [ControllerContext](#).

Definition

```
#include <controller.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class ControllerContext {
                virtual void update(void)=0;
            }
        }
    }
}
```

Return Values

None

Description

This is the update function of [ControllerContext](#). `scePadGetData()` is called within this function and the data of the controller in [ControllerContext](#) is updated; thus, call this function when the data needs to be updated.

isButtonDown

Function to determine whether a button is being pressed.

Definition

```
#include <controller.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class ControllerContext {
                virtual bool isButtonDown(
                    uint32_t buttons,
                    ButtonEventPattern pattern =
kButtonEventPatternAll
                )=0 const;
            }
        }
    }
}
```

Arguments

buttons
pattern

The enum value of Button used to judge the function
The matching pattern of the button. PATTERN_ALL is specified as the default parameter.

Return Values

Value	Description
true	The button specified in buttons is being pressed in reference to the pattern specified by pattern
false	The button specified in buttons is not being pressed in reference to the pattern specified by pattern

Description

This function refers to the matching pattern specified in the pattern argument to determine whether the button specified in the buttons argument is currently being pressed.

SCE CONFIDENTIAL

isButtonUp

Function to determine that a button is not currently being pressed.

Definition

```
#include <controller.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class ControllerContext {
                virtual bool isButtonUp(
                    uint32_t buttons,
                    ButtonEventPattern pattern =
kButtonEventPatternAll
                )=0 const;
            }
        }
    }
}
```

Arguments

buttons
pattern

The enum value of Button used to judge the function
The matching pattern of the button. PATTERN_ALL is specified as the default parameter.

Return Values

Value	Description
true	The button specified in buttons is not being pressed in reference to the pattern specified by pattern
false	The button specified in buttons is being pressed in reference to the pattern specified by pattern

Description

This function refers to the matching pattern specified in the pattern argument to determine whether the button specified in the buttons argument is currently not being pressed.

isButtonPressed

Function to determine that a button has been pressed.

Definition

```
#include <controller.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class ControllerContext {
                virtual bool isButtonPressed(
                    uint32_t buttons,
                    ButtonEventPattern pattern =
kButtonEventPatternAll
                )=0 const;
            }
        }
    }
}
```

Arguments

buttons
pattern

The enum value of Button used to judge the function
The matching pattern of the button. PATTERN_ALL is specified as the default parameter.

Return Values

Value	Description
true	The button specified in buttons has been pressed in reference to the pattern specified by pattern
false	The button specified in buttons has not been pressed in reference to the pattern specified by pattern

Description

This function refers to the matching pattern specified in the pattern argument for the button specified in the buttons arguments and compares the controller values obtained with the last [update \(\)](#) call and the [update \(\)](#) call preceding it to determine whether the button has been pressed.

isButtonReleased

Function to determine that a button has been released.

Definition

```
#include <controller.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class ControllerContext {
                virtual bool isButtonReleased(
                    uint32_t buttons,
                    ButtonEventPattern pattern =
kButtonEventPatternAll
                )=0 const;
            }
        }
    }
}
```

Arguments

buttons
pattern

The enum value of Button used to judge the function
The matching pattern of the button. PATTERN_ALL is specified as the default parameter.

Return Values

Value	Description
true	The button specified in buttons has been released in reference to the pattern specified by pattern
false	The button specified in buttons has not been released in reference to the pattern specified by pattern

Description

This function refers to the matching pattern specified in the pattern argument for the button specified in the buttons arguments and compares the controller values obtained with the last [update \(\)](#) call and the [update \(\)](#) call preceding it to determine whether the button has been released.

SCE CONFIDENTIAL

getLeftStick

Function to get left analog stick data.

Definition

```
#include <controller.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class ControllerContext {
                virtual const
sce::Vectormath::Simd::Aos::Vector2 &getLeftStick(void)=0 const;
            }
        }
    }
}
```

Return Values

Value	Description
sce::Vectormath::Simd::Aos::Vector2&	Returns the value of the left analog stick in the Vector2 type. In the X and Y values of Vector2, the X direction and Y direction of the left analog stick are stored in a float value, respectively.

Description

This is the function to get left analog stick data. In the X and Y values of Vector2 of the return value, the X direction and Y direction of the left analog stick are stored in a float value, respectively. In this function, the dead zones are also taken into account and the dead zones can be changed with [setDeadZone \(\)](#).

getRightStick

Function to get right analog stick data.

Definition

```
#include <controller.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class ControllerContext {
                virtual const
sce::Vectormath::Simd::Aos::Vector2 &getRightStick(void)=0 const;
            }
        }
    }
}
```

Return Values

Value	Description
sce::Vectormath::Simd::Aos::Vector2&	Returns the value of the right analog stick in the Vector2 type. In the X and Y values of Vector2, the X direction and Y direction of the right analog stick are stored in a float value, respectively.

Description

This is the function to get right analog stick data. In the X and Y values of Vector2 of the return value, the X direction and Y direction of the right analog stick are stored in a float value, respectively. In this function, the dead zones are also taken into account and the dead zones can be changed with [setDeadZone \(\)](#).

SCE CONFIDENTIAL

setDeadZone

Function to set analog stick dead zones.

Definition

```
#include <controller.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class ControllerContext {
                virtual void setDeadZone(
                    float deadZone
                )=0;
            }
        }
    }
}
```

Arguments

deadZone The dead zone value of analog stick

Return Values

None

Description

This is the function to set the dead zones of analog stick. The default value is set to 32.0f/127.0f (approximately 0.2520f), and this value is derived from the dead zone of analog sticks. For details on the dead zone, refer to the "System Software Overv" and "Controller Service Overview" documents.

getData

Function to obtain the controller data currently held by [ControllerContext](#).

Definition

```
#include <controller.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class ControllerContext {
                virtual int getData(
                    uint32_t port,
                    ControllerData *data,
                    uint32_t length
                )=0 const;
            }
        }
    }
}
```

Arguments

<i>port</i>	Port number
<i>data</i>	Array of the write destination of the ControllerData structure
<i>length</i>	Length of the data array

Return Values

Returns the number of written data items for success. Returns an error code for failures.

Description

This function obtains the controller data array currently held by [ControllerContext](#). This is the buffer count designated at the time of initialization for maximum array length.

SCE CONFIDENTIAL

sce::SampleUtil::Input::MotionContextD

ata

000004892117

Summary

sce::SampleUtil::Input::MotionContextData

Structure that handles device states and motion sensor values.

Definition

```
#include <motion.h>
struct MotionContextData { };
```

Description

This structure handles device states and motion sensor values. For details on device coordinate systems, refer to the "libmotion Overview" document.

Fields

Public Instance Fields

uint32_t <i>deviceTimestamp</i>	Device timestamp of sensor device (microseconds)
uint64_t <i>systemTimestamp</i>	Timestamp for time when data packet was received from host (microseconds)
sce::Vectormath::Simd::Aos::Quat <i>relativeOrientation</i>	Device orientation expressed in quaternions <x, y, z, w>
sce::Vectormath::Simd::Aos::Vector3 <i>linearAcceleration</i>	Accelerometer reading <x, y, z> (G)
sce::Vectormath::Simd::Aos::Vector3 <i>angularVelocity</i>	Gyro sensor reading <x, y, z> (rad/s)
sce::Vectormath::Simd::Aos::Vector3 <i>userInterfaceOrientation</i>	Basic device orientation based on gravity<x, y, z>
sce::Vectormath::Simd::Aos::Matrix4 <i>northEastDownOrientation</i>	4x4 rotation matrix for orientation based on NED (North East Down)
uint8_t <i>magneticFieldStability</i>	Value that indicates magnetic field stability.

SCE CONFIDENTIAL

sce::SampleUtil::Input::MotionContext

000004892117

Summary

sce::SampleUtil::Input::MotionContext

Class that handles motion sensor data.

Definition

```
#include <motion.h>
class MotionContext : public sce::SampleUtil::Resource {};
```

Description

[MotionContext](#) is a class that handles motion sensor values and device orientation values calculated from the motion sensor.

Methods Summary

Methods	Description
update	MotionContext class update.
enableFunctions	Function flag settings.
getEnabledFunctions	Gets the function flag.
getMotionState	Get current device state and motion sensor data.
getMotionContextData	Get current device state and motion sensor data.
resetReferenceOrientation	Reset device orientation.
rotateReferenceOrientationYaw	Device yaw rotation.
setUserInterfaceOrientationThreshold	Set angle threshold value used when determining basic orientation output.
enableAccelerometerTiltCorrection	Set accelerometer tilt correction filter.
enableGyroDeadband	Set gyro dead-banding filter.
enableGyroBiasCorrection	Enable/disable gyro bias correction filter.

Public Instance Methods

update

[MotionContext](#) class update.

Definition

```
#include <motion.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class MotionContext {
                virtual int update(void)=0;
            }
        }
    }
}
```

Return Values

Value	Description
(>=0)	Success
(<0)	Failure

Description

This updates the [MotionContext](#) class.

SCE CONFIDENTIAL

enableFunctions

Function flag settings.

Definition

```
#include <motion.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class MotionContext {
                virtual int enableFunctions(
                    uint32_t functionFlag
                )=0;
            }
        }
    }
}
```

Arguments

functionFlag Function flag

Return Values

Value	Description
(≥ 0)	Success
(<0)	Failure

Description

This is for enabling or disabling each feature for the [MotionContext](#). The argument functionFlag is treated as a bit pattern. For functionFlag specification, add the required items from the MotionContextFunctionFlag definition with an OR operation. When handling the magnetometer, kMotionContextFunctionTypeAccelerometerAndGyro is required in addition to kMotionContextFunctionTypeMagnetometer.

SCE CONFIDENTIAL

getEnabledFunctions

Gets the function flag.

Definition

```
#include <motion.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class MotionContext {
                virtual uint32_t getEnabledFunctions(void)=0
            const;
        }
    }
}
```

Return Values

Value	Description
uint32_t	Function flag

Description

This gets the current function flag set values.

SCE CONFIDENTIAL

getMotionState

Get current device state and motion sensor data.

Definition

```
#include <motion.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class MotionContext {
                virtual int getMotionState(
                    SceMotionState *motionState
                )=0 const;
            }
        }
    }
}
```

Arguments

motionState Pointer to SceMotionState structure

Return Values

Value	Description
(≥ 0)	Success
(<0)	Failure

Description

This obtains the current motion sensor data and device state calculated from the motion sensor.

SCE CONFIDENTIAL

getMotionContextData

Get current device state and motion sensor data.

Definition

```
#include <motion.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class MotionContext {
                virtual int getMotionContextData(
                    MotionContextData *motionContextData
                )=0 const;
            }
        }
    }
}
```

Arguments

motionContextData Pointer to [MotionContextData](#) structure

Return Values

Value	Description
(≥ 0)	Success
(<0)	Failure

Description

This obtains the current motion sensor data and device state calculated from the motion sensor.

SCE CONFIDENTIAL

resetReferenceOrientation

Reset device orientation.

Definition

```
#include <motion.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class MotionContext {
                virtual int resetReferenceOrientation()=0;
            }
        }
    }
}
```

Return Values

Value	Description
(>=0)	Success
(<0)	Failure

Description

This function resets the device orientation. Even if a reset is called, the filter settings and sampling mode will not change. For the behavior during a reset, refer to the "Using the Library" chapter of the "libmotion Overview" document.

rotateReferenceOrientationYaw

Device yaw rotation.

Definition

```
#include <motion.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class MotionContext {
                virtual int rotateReferenceOrientationYaw(
                    float radians
                )=0;
            }
        }
    }
}
```

Arguments

radians Yaw rotation angle

Return Values

Value	Description
(≥ 0)	Success
(<0)	Failure

Description

When this function is used, the device's yaw can be corrected with yaw rotation.

SCE CONFIDENTIAL

setUserInterfaceOrientationThreshold

Set angle threshold value used when determining basic orientation output.

Definition

```
#include <motion.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class MotionContext {
                virtual int
                setUserInterfaceOrientationThreshold(
                    float radians
                )=0;
            }
        }
    }
}
```

Arguments

<i>radians</i>	Angle that sets the threshold value (degrees). Settings in a range from 0 to 45 degrees are available.
----------------	--

Return Values

Value	Description
(≥ 0)	Success
(<0)	Failure

Description

This sets the angle threshold value used when determining basic orientation output. For details on the angle threshold value, refer to the "libmotion Overview" document.

SCE CONFIDENTIAL

enableAccelerometerTiltCorrection

Set accelerometer tilt correction filter.

Definition

```
#include <motion.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class MotionContext {
                virtual void enableAccelerometerTiltCorrection(
                    bool enable
                )=0;
            }
        }
    }
}
```

Arguments

enable Enables or disables. Enabled if it is true. The default is true

Return Values

None

Description

This sets the accelerometer tilt correction filter.

SCE CONFIDENTIAL

enableGyroDeadband

Set gyro dead-banding filter.

Definition

```
#include <motion.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class MotionContext {
                virtual void enableGyroDeadband(
                    bool enable
                )=0;
            }
        }
    }
}
```

Arguments

enable Enables or disables. Enabled if it is true. The default is true

Return Values

None

Description

This sets the gyro dead-banding filter.

enableGyroBiasCorrection

Enable/disable gyro bias correction filter.

Definition

```
#include <motion.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class MotionContext {
                virtual void enableGyroBiasCorrection(
                    bool enable
                )=0;
            }
        }
    }
}
```

Arguments

enable Enables or disables. Enabled if it is true. The default is true

Return Values

None

Description

This function enables/disables the gyro bias correction filter. This function can be used even if sampling has not started. When the gyro bias correction filter is disabled, note that output from libmotion will drift. Therefore, the gyro bias correction filter is enabled by default in libmotion, and it is strongly recommended that this is not changed. For details on the gyro bias correction filter, refer to the "Using the Library" chapter of the "libmotion Overview" document.

sce::SampleUtil::Input::TouchContextOption

Summary

sce::SampleUtil::Input::TouchContextOption

Structure for setting [TouchContext](#).

Definition

```
#include <touch.h>
struct TouchContextOption {};
```

Description

This is the structure to handle the set value of [TouchContext](#). Used for the argument used when the initialize of [TouchContext](#) is executed.

Fields

Public Instance Fields

uint32_t numBufs

Number of buffers to receive touch input data.
This sets the number of buffers to receive touch input data for each touch port. The initial values of SCE_TOUCH_PORT_FRONT and SCE_TOUCH_PORT_BACK are 1 and 0, respectively. For details on touch input data, refer to the "Touch Service Reference" document.

sce::SampleUtil::Input::TouchPrimitive GestureEvent

Summary

sce::SampleUtil::Input::TouchPrimitiveGestureEvent

Structure for getting the primitive gesture event.

Definition

```
#include <touch.h>
struct TouchPrimitiveGestureEvent { };
```

Description

This is the structure used for getting the primitive gesture event data.

Fields

Public Instance Fields

TouchPort port	Touch panel port.
SceSystemGesturePrimitiveTouchEvent event	Primitive gesture event structure For details, refer to the "libsystemgesture Reference" document.

sce::SampleUtil::Input::TouchGestureEvent

Summary

sce::SampleUtil::Input::TouchGestureEvent

Structure for getting the gesture event.

Definition

```
#include <touch.h>
struct TouchGestureEvent { };
```

Description

This is the structure used for getting the gesture event data.

Fields

Public Instance Fields

TouchPort <i>port</i>	Touch panel port.
SceSystemGestureTouchEvent <i>event</i>	Gesture event structure For details, refer to the "libsystemgesture Reference" document.

SCE CONFIDENTIAL

sce::SampleUtil::Input::TouchContext

000004892117

Summary

sce::SampleUtil::Input::TouchContext

Class to handle input data from the touch panel.

Definition

```
#include <touch.h>
class TouchContext : public sce::SampleUtil::Resource {};
```

Description

This reads input data from the touch panel and recognizes the consecutive touch operations. For details on touch input data, refer to the "Touch Service Reference" document. For details on consecutive touch operations, refer to the "libsystemgesture Reference" document.

Methods Summary

Methods	Description
<u>update</u>	Updates the TouchContext class.
<u>setFunctionFlag</u>	Function flag settings.
<u>getFunctionFlag</u>	Gets the function flag.
<u>getRawData</u>	Gets touch panel input data.
<u>getPanelInfo</u>	Gets touch panel information.
<u>getNumberOfPrimitiveGestureEvents</u>	Gets the primitive gesture event count.
<u>getPrimitiveGestureEvents</u>	Gets the primitive gesture event.
<u>getNumberOfGestureEvents</u>	Gets the gesture event count.
<u>getGestureEvents</u>	Gets gesture events.

Public Instance Methods

update

Updates the [TouchContext](#) class.

Definition

```
#include <touch.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class TouchContext {
                virtual int update(void)=0;
            }
        }
    }
}
```

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_NOT_INITIALIZED	TouchContext is not initialized
SCE_SAMPLE_UTIL_ERROR_NO_RAW_DATA_BUFFER	No touch input data buffer
<SCE_OK	Failure (Error codes of libsystemgesture and touch service)

Description

This updates the [TouchContext](#) class.

SCE CONFIDENTIAL

setFunctionFlag

Function flag settings.

Definition

```
#include <touch.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class TouchContext {
                virtual int setFunctionFlag(
                    const TouchPort port,
                    const uint32_t functionFlag
                )=0;
            }
        }
    }
}
```

Arguments

<i>port</i>	Touch panel port number
<i>functionFlag</i>	Function flag

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_NO_RAW_DATA_BUFFER	No touch input data buffer
SCE_SAMPLE_UTIL_ERROR_NOT_INITIALIZED	TouchContext is not initialized
<SCE_OK	Failure (Error codes of touch service)

Description

This is for enabling or disabling each function for the [TouchContext](#). The argument functionFlag is treated as a bit pattern. For functionFlag specification, add the required items from the TouchContextFunctionFlag definition with an OR operation.

SCE CONFIDENTIAL

getFunctionFlag

Gets the function flag.

Definition

```
#include <touch.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class TouchContext {
                virtual uint32_t getFunctionFlag(
                    const TouchPort port
                )=0 const;
            }
        }
    }
}
```

Arguments

port Touch panel port number

Return Values

Value	Description
uint32_t	Function flag

Description

This gets the current function flag set values.

SCE CONFIDENTIAL

getRawData

Gets touch panel input data.

Definition

```
#include <touch.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class TouchContext {
                virtual int getRawData(
                    const TouchPort port,
                    SceTouchData *touchData,
                    const uint32_t size
                )=0 const;
            }
        }
    }
}
```

Arguments

<i>port</i>	Touch panel port number
<i>touchData</i>	Pointer to the touch panel input data structure
<i>size</i>	Number of touch panel input data items received

Return Values

Value	Description
0	or greater Number of touch panel input data items obtained (success)
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid
SCE_SAMPLE_UTIL_ERROR_NO_RAW_DATA_BUFFER	No touch input data buffer
SCE_SAMPLE_UTIL_ERROR_FUNCTION_NOT_ENABLED	Function flag is not enabled

Description

This gets the data input from touch panel. For details on SceTouchData, refer to the "Touch Service Reference" document.

SCE CONFIDENTIAL

getPanelInfo

Gets touch panel information.

Definition

```
#include <touch.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class TouchContext {
                virtual int getPanelInfo(
                    const TouchPort port,
                    SceTouchPanelInfo *panelInfo
                )=0 const;
            }
        }
    }
}
```

Arguments

<i>port</i>	Touch panel port number
<i>panelInfo</i>	Pointer to SceTouchPanelInfo

Return Values

Value	Description
SCE_OK	Success
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid

Description

This gets touch panel information. For details on SceTouchPanelInfo, refer to the "Touch Service Reference" document.

SCE CONFIDENTIAL

getNumberOfPrimitiveGestureEvents

Gets the primitive gesture event count.

Definition

```
#include <touch.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class TouchContext {
                virtual int
            getNumberOfPrimitiveGestureEvents()=0 const;
            }
        }
    }
}
```

Return Values

Value	Description
0	or greater Primitive gesture event count (success)
SCE_SAMPLE_UTIL_ERROR_FUNCTION_NOT_ENABLED	Function flag is not enabled

Description

This gets the total number of primitive gesture events. For details on primitive gesture events, refer to the "libsystemgesture Reference" document.

SCE CONFIDENTIAL

getPrimitiveGestureEvents

Gets the primitive gesture event.

Definition

```
#include <touch.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class TouchContext {
                virtual int getPrimitiveGestureEvents(
                    TouchPrimitiveGestureEvent *events,
                    const uint32_t size
                )=0 const;
            }
        }
    }
}
```

Arguments

<code>events</code>	Pointer to a primitive gesture event structure
<code>size</code>	Number of primitive gesture events received

Return Values

Value	Description
0	or greater Primitive gesture event count obtained (success)
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid
SCE_SAMPLE_UTIL_ERROR_NOT_INITIALIZED	<code>TouchContext</code> is not initialized
SCE_SAMPLE_UTIL_ERROR_FUNCTION_NOT_ENABLED	Function flag is not enabled

Description

This gets the primitive gesture events. For details on primitive gesture events, refer to the "libsystemgesture Reference" document.

SCE CONFIDENTIAL

getNumberOfGestureEvents

Gets the gesture event count.

Definition

```
#include <touch.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class TouchContext {
                virtual int getNumberOfGestureEvents()=0 const;
            }
        }
    }
}
```

Return Values

Value	Description
0	or greater Gesture event count (success)
SCE_SAMPLE_UTIL_ERROR_FUNCTION_NOT_ENABLED	Function flag is not enabled

Description

This gets the total number of gesture events. For details on gesture events, refer to the "libsystemgesture Reference" document.

SCE CONFIDENTIAL

getGestureEvents

Gets gesture events.

Definition

```
#include <touch.h>
namespace sce {
    namespace SampleUtil {
        namespace Input {
            class TouchContext {
                virtual int getGestureEvents(
                    TouchGestureEvent *events,
                    const uint32_t size
                )=0 const;
            }
        }
    }
}
```

Arguments

<code>events</code>	Pointer to a gesture event structure
<code>size</code>	Number of gesture events received

Return Values

Value	Description
0	or greater Gesture event count obtained (success)
SCE_SAMPLE_UTIL_ERROR_INVALID_PARAM	Parameter invalid
SCE_SAMPLE_UTIL_ERROR_NOT_INITIALIZED	<code>TouchContext</code> is not initialized
SCE_SAMPLE_UTIL_ERROR_FUNCTION_NOT_ENABLED	Function flag is not enabled

Description

This gets gesture events. For details on gesture events, refer to the "libsystemgesture Reference" document.

SCE CONFIDENTIAL

sce::SampleUtil::Resource

000004892117

Summary

sce::SampleUtil::Resource

[Resource](#) base class.

Definition

```
#include <sampleutil_common.h>
class Resource {};
```

Description

This is the base class for resources generated within the library.

SCE CONFIDENTIAL

sce::SampleUtil::SampleSkeleton

000004892117

Summary

sce::SampleUtil::SampleSkeleton

Sample skeleton.

Definition

```
#include <sampleutil.h>
class SampleSkeleton {};
```

Description

This is the sample skeleton which has a sample utility function.

Methods Summary

Methods	Description
SampleSkeleton	Constructor.
~SampleSkeleton	Destructor.
initialize	Initialization.
update	Update.
render	Drawing.
finalize	Termination processing.
initializeUtil	Initializes a utility.
updateUtil	Update utility.
finalizeUtil	Termination processing of a utility.
renderDebugText	Draw debug text.
getUserIDManager	Get user ID manager.
getGraphicsContext	Get graphics context.
getControllerContext	Get controller context.
getPadContextOfInitialUser	Get pad context.
getSpriteRenderer	Get 2D sprite rendering context.
getMotionContext	Get motion context.
getTouchContext	Get touch context.
getAudioContext	Get audio context.

Inner Classes, Structures, and Namespaces

Item	Description
sce::SampleUtil::SampleSkeleton::SampleSkeletonOption	Sample skeleton option.

Enumerated Types

SampleSkeletonFunctionFlag

The enum value of the flag of the function to be used in [SampleSkeleton](#).

Definition

```
#include <skeleton.h>
namespace sce {
    namespace SampleUtil {
        class SampleSkeleton {
            enum SampleSkeletonFunctionFlag {
                kFunctionFlagGraphics = (1<<0),
                kFunctionFlagController = (1<<1),
                kFunctionFlagSpriteRenderer = (1<<2),
                kFunctionFlagDebugFont = (1<<3),
                kFunctionFlagMotion = (1<<4),
                kFunctionFlagTouch = (1<<5),
                kFunctionFlagAudio = (1<<6),
                kFunctionFlagFios2 = (1<<7),
                kFunctionFlagUserIdManager = (1<<8),
                kFunctionFlagDefault = (1<<0) | (1<<1) | (1<<2) |
                (1<<3) | (1<<8),
                kFunctionFlagAll = (1<<0) | (1<<1) | (1<<2) | (1<<3)
                | (1<<4) | (1<<5) | (1<<6) | (1<<7) | (1<<8)
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kFunctionFlagGraphics	(1<<0)	Graphics function flag value.
kFunctionFlagController	(1<<1)	Controller function flag value.
kFunctionFlagSpriteRenderer	(1<<2)	2D sprite rendering function flag value
kFunctionFlagDebugFont	(1<<3)	Debug font function flag value.
kFunctionFlagMotion	(1<<4)	Motion function flag value.
kFunctionFlagTouch	(1<<5)	Touch drawing function flag value.
kFunctionFlagAudio	(1<<6)	Audio function flag value.
kFunctionFlagFios2	(1<<7)	FIOS2 function flag value.
kFunctionFlagUserIdManager	(1<<8)	User ID Manager feature flag value.
kFunctionFlagDefault	(1<<0) (1<<1) (1<<2) (1<<3) (1<<8)	Default function (graphics function, controller function, 2D sprite rendering function, debug font function) flag value.

SCE CONFIDENTIAL

Macro	Value	Description
kFunctionFlagAll	(1<<0)	All function flag value.
	(1<<1)	
	(1<<2)	
	(1<<3)	
	(1<<4)	
	(1<<5)	
	(1<<6)	
	(1<<7)	
	(1<<8)	

Description

Specify this for [initializeUtil\(\)](#).

SCE CONFIDENTIAL

ControllerFunctionContextMode

Controller feature context mode.

Definition

```
#include <skeleton.h>
namespace sce {
    namespace SampleUtil {
        class SampleSkeleton {
            enum ControllerFunctionContextMode {
                kControllerContextMode = 0,
                kPadContextMode
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kControllerContextMode	0	Controller context mode.
kPadContextMode	N/A	Pad context mode.

Description

This is the controller feature context mode.

Constructors and Destructors

SampleSkeleton

Constructor.

Definition

```
#include <skeleton.h>
namespace sce {
    namespace SampleUtil {
        class SampleSkeleton {
            SampleSkeleton(void);
        };
    }
}
```

Return Values

None

Description

This is a constructor.

SCE CONFIDENTIAL

~SampleSkeleton

Destructor.

Definition

```
#include <sample.h>
namespace sce {
    namespace SampleUtil {
        class SampleSkeleton {
            virtual ~SampleSkeleton(void);
        };
    }
}
```

Return Values

None

Description

This is a destructor.

Public Instance Methods

initialize

Initialization.

Definition

```
#include <skelton.h>
namespace sce {
    namespace SampleUtil {
        class SampleSkeleton {
            virtual int initialize(void)=0;
        }
    }
}
```

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This function defines the initialization processing. This is overridden by an application class for each sample.

SCE CONFIDENTIAL

update

Update.

Definition

```
#include <skelton.h>
namespace sce {
    namespace SampleUtil {
        class SampleSkeleton {
            virtual int update(void)=0;
        }
    }
}
```

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This function defines the update processing. This is overridden by an application class for each sample.

SCE CONFIDENTIAL

render

Drawing.

Definition

```
#include <skelton.h>
namespace sce {
    namespace SampleUtil {
        class SampleSkeleton {
            virtual void render(void)=0;
        }
    }
}
```

Return Values

None

Description

This function defines the render processing. This is overridden by an application class for each sample.

SCE CONFIDENTIAL

finalize

Termination processing.

Definition

```
#include <skelton.h>
namespace sce {
    namespace SampleUtil {
        class SampleSkeleton {
            virtual int finalize(void)=0;
        }
    }
}
```

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This function defines the termination processing. This is overridden by an application class for each sample.

Protected Instance Methods

initializeUtil

Initializes a utility.

Definition

```
#include <skeleton.h>
namespace sce {
    namespace SampleUtil {
        class SampleSkeleton {
            int initializeUtil(
                uint32_t functionFlags = kFunctionFlagDefault,
                uint32_t width = 960,
                uint32_t height = 544,
                SampleSkeletonOption *option = NULL
            );
        };
    }
}
```

Arguments

<i>functionFlags</i>	Flag value for the function to use
<i>width</i>	Width of the main render target displayed on the display
<i>height</i>	Height of the main render target displayed on the display
<i>option</i>	Pointer to the SampleSkeletonOption structure. This is initialized by the default value if NULL is specified. The controller context mode is the default for the controller feature modes. kFunctionFlagUserIdManager is required to use the pad context mode in the controller feature modes.

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This initializes each function of the utility in accordance with functionFlags. For use of the 2D sprite rendering feature (kFunctionFlagSpriteRenderer), the graphics function (kFunctionFlagGraphics) is required. For streaming with the audio function (kFunctionFlagAudio), the FIOS2 function (kFunctionFlagFios2) is required.

SCE CONFIDENTIAL

updateUtil

Update utility.

Definition

```
#include <skelton.h>
namespace sce {
    namespace SampleUtil {
        class SampleSkeleton {
            virtual int updateUtil(void);
        }
    }
}
```

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This updates a utility.

SCE CONFIDENTIAL

finalizeUtil

Termination processing of a utility.

Definition

```
#include <skelton.h>
namespace sce {
    namespace SampleUtil {
        class SampleSkeleton {
            virtual int finalizeUtil(void);
        }
    }
}
```

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This executes termination processing of a utility.

SCE CONFIDENTIAL

renderDebugText

Draw debug text.

Definition

```
#include <skeleton.h>
namespace sce {
    namespace SampleUtil {
        class SampleSkeleton {
            virtual void renderDebugText(void);
        }
    }
}
```

Return Values

None

Description

This draws a debug text. This is overridden by an application for each sample.

SCE CONFIDENTIAL

getUserIdManager

Get user ID manager.

Definition

```
#include <skeleton.h>
namespace sce {
    namespace SampleUtil {
        class SampleSkeleton {
            sce::SampleUtil::System::UserIdManager
        *getUserIdManager(void);
    }
}
```

Return Values

User ID manager instance

Description

This gets the user ID manager.

SCE CONFIDENTIAL

getGraphicsContext

Get graphics context.

Definition

```
#include <skelton.h>
namespace sce {
    namespace SampleUtil {
        class SampleSkeleton {
            sce::SampleUtil::Graphics::GraphicsContext
            *getGraphicsContext(void);
        }
    }
}
```

Return Values

[Graphics](#) context

Description

This gets a graphics context. If kFunctionFlagGraphics is not specified at the time of initialization, NULL will be returned.

SCE CONFIDENTIAL

getControllerContext

Get controller context.

Definition

```
#include <skeleton.h>
namespace sce {
    namespace SampleUtil {
        class SampleSkeleton {
            sce::SampleUtil::Input::ControllerContext
            *getControllerContext(void);
        }
    }
}
```

Return Values

Controller context

Description

This gets a controller context. If kFunctionFlagController is not specified at the time of initialization, NULL will be returned.

SCE CONFIDENTIAL

getPadContextOfInitialUser

Get pad context.

Definition

```
#include <skeleton.h>
namespace sce {
    namespace SampleUtil {
        class SampleSkeleton {
            sce::SampleUtil::Input::PadContext
            *getPadContextOfInitialUser(void);
        }
    }
}
```

Return Values

Pad context

Description

This gets a pad context. If kFunctionFlagController is not specified at the time of initialization, NULL will be returned.

SCE CONFIDENTIAL

getSpriteRenderer

Get 2D sprite rendering context.

Definition

```
#include <skeleton.h>
namespace sce {
    namespace SampleUtil {
        class SampleSkeleton {
            sce::SampleUtil::Graphics::SpriteRenderer
*getSpriteRenderer(void);
        }
    }
}
```

Return Values

2D sprite rendering context

Description

This gets a 2D sprite rendering context. If kFunctionFlagSpriteRenderer is not specified at the time of initialization, NULL will be returned.

SCE CONFIDENTIAL

getMotionContext

Get motion context.

Definition

```
#include <skeleton.h>
namespace sce {
    namespace SampleUtil {
        class SampleSkeleton {
            sce::SampleUtil::Input::MotionContext
        *getMotionContext(void);
    }
}
```

Return Values

Motion context.

Description

This gets a motion context. If kFunctionFlagMotion is not specified at the time of initialization, NULL will be returned.

SCE CONFIDENTIAL

getTouchContext

Get touch context.

Definition

```
#include <skeleton.h>
namespace sce {
    namespace SampleUtil {
        class SampleSkeleton {
            sce::SampleUtil::Input::TouchContext
            *getTouchContext(void);
        }
    }
}
```

Return Values

Touch context

Description

This gets a touch context. If kFunctionFlagTouch is not specified at the time of initialization, NULL will be returned.

SCE CONFIDENTIAL

getAudioContext

Get audio context.

Definition

```
#include <skeleton.h>
namespace sce {
    namespace SampleUtil {
        class SampleSkeleton {
            sce::SampleUtil::Audio::AudioContext
        *getAudioContext(void);
    }
}
```

Return Values

[Audio](#) context

Description

This gets an audio context. If kFunctionFlagAudio is not specified at the time of initialization, NULL will be returned.

sce::SampleUtil::SampleSkeleton::SampleSkeletonOption

Summary

sce::SampleUtil::SampleSkeleton::SampleSkeletonOption

Sample skeleton option.

Definition

```
#include <sample.h>
struct SampleSkeletonOption { };
```

Description

This is the sample skeleton option.

Fields

Public Instance Fields

```
Graphics::GraphicsContextOption
*graphicsOption
Input::ControllerContextOption
*controllerOption
uint32_t fontSize

Audio::AudioContextOption
*audioOption
Input::TouchContextOption
*touchOption
SceFiosParams *fios2Option
ControllerFunctionContextMode
controllerFunctionContextMode
Input::PadContextOption
*padOption
```

[Graphics](#) option.

This is the controller option.

This is the value to be set for font size mode for the debug font. The value must be SCE_DBGFONT_FONTSIZE_DEFAULT(0) or SCE_DBGFONT_FONTSIZE_LARGE(1).

[Audio](#) option.

Touch option.

FIOS2 option.

This is the controller feature context mode.

PadContext option.

Methods Summary

Methods	Description
SampleSkeletonOption	Constructor.

Constructors and Destructors

SampleSkeletonOption

Constructor.

Definition

```
#include <sample.h>
SampleSkeletonOption(void);
```

Return Values

None

Description

This is a constructor.

SCE CONFIDENTIAL

sce::SampleUtil::System

000004892117

Summary

sce::SampleUtil::System

System-associated definitions.

Definition

```
namespace System {}
```

Description

These are the System-associated definitions.

Function Summary

Function	Description
createUserIdManager	Initialization function of UserIdManager .
createSaveData	Create save data class instance.

Inner Classes, Structures, and Namespaces

Item	Description
sce::SampleUtil::System::UserIdList	Structure that stores the user ID list.
sce::SampleUtil::System::UserLoginStatusChangedEvents	Structure that stores the user login status (logged in/logged out) change events.
sce::SampleUtil::System::UserIdManager	Class for managing user IDs.
sce::SampleUtil::System::UserIdManagerOption	Options structure for UserIdManager .
sce::SampleUtil::System::SaveDataSlotParam	Save data slot parameter structure.
sce::SampleUtil::System::SaveDataSearchCond	Save data slot search condition structure.
sce::SampleUtil::System::SaveDataSearchResult	Save data slot search result structure.
sce::SampleUtil::System::DataSaveItem	Structure for specifying target save data to save.
sce::SampleUtil::System::SaveDataMountPoint	Save data mount point structure.
sce::SampleUtil::System::DataRemoveItem	Structure for specifying target save data to remove.
sce::SampleUtil::System::SaveData	Save data.
sce::SampleUtil::System::SaveDataOption	Structure for initializing SaveData .

Enumerated Types

UserColor

Enumerator type for user colors.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            enum UserColor {
                kUserColorBlue = 0,
                kUserColorRed = 1,
                kUserColorGreen = 2,
                kUserColorPink = 3
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kUserColorBlue	0	Blue.
kUserColorRed	1	Red.
kUserColorGreen	2	Green.
kUserColorPink	3	Pink.

Description

This is the enumerator type for user colors.

SCE CONFIDENTIAL

SaveDataSlotStatus

Save data slot status enumerated type.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            enum SaveDataSlotStatus {
                kSaveDataSlotStatusAvailable = 0,
                kSaveDataSlotStatusBroken = 1
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kSaveDataSlotStatusAvailable	0	Normal status.
kSaveDataSlotStatusBroken	1	Broken status.

Description

This is the save data slot status enumerated type.

SCE CONFIDENTIAL

SaveDataSlotSortKey

Save data slot search sort key enumerated type.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            enum SaveDataSlotSortKey {
                kSaveDataSlotSortKeySlotId = 0,
                kSaveDataSlotSortKeyUserParam = 1,
                kSaveDataSlotSortKeySize = 2,
                kSaveDataSlotSortKeyMtime = 3
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kSaveDataSlotSortKeySlotId	0	Use the save data slot ID as the sort key.
kSaveDataSlotSortKeyUserParam	1	Use the user parameter as the sort key.
kSaveDataSlotSortKeySize	2	Use the data size as the sort key.
kSaveDataSlotSortKeyMtime	3	Use the last modified date/time as the sort key.

Description

This is the save data slot search sort key enumerated type.

SCE CONFIDENTIAL

SaveDataSlotSortOrder

Save data slot search sort order enumerated type.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            enum SaveDataSlotSortOrder {
                kSaveDataSlotSortOrderAscent = 0,
                kSaveDataSlotSortOrderDescent = 1
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kSaveDataSlotSortOrderAscent	0	Sort in ascending order.
kSaveDataSlotSortOrderDescent	1	Sort in descending order.

Description

This is the save data slot search sort order enumerated type.

SCE CONFIDENTIAL

SaveDataSaveMode

Save data save mode enumerated type.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            enum SaveDataSaveMode {
                kSaveDataSaveModeFile = 0,
                kSaveDataSaveModeFileTruncate = 1,
                kSaveDataSaveModeDirectory = 2
            };
        }
    }
}
```

Enumeration Values

Macro	Value	Description
kSaveDataSaveModeFile	0	File operation.
kSaveDataSaveModeFileTruncate	1	File operation (truncate file with the specified offset and size)
kSaveDataSaveModeDirectory	2	Directory operation.

Description

This is the save data save mode enumerated type.

Type Definitions

UserId

User ID type.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            typedef int UserId;
        }
    }
}
```

Description

This is the user ID type.

SCE CONFIDENTIAL

UserIdManagerOption

Options structure for [UserIdManager](#).

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            typedef
            struct sce::SampleUtil::System::UserIdManagerOption {
                } UserIdManagerOption;
            }
        }
}
```

Description

This is an options structure for [UserIdManager](#). This is used by specifying it to the option argument of [createUserIdManager\(\)](#).

SCE CONFIDENTIAL

SaveDataSlotId

Save data slot ID type.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            typedef uint32_t SaveDataSlotId;
        }
    }
}
```

Description

This is the save data slot ID type.

SCE CONFIDENTIAL

SaveDataSlotParam

Save data slot parameter structure.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            typedef struct sce::SampleUtil::System::SaveDataSlotParam {
                SaveDataSlotStatus status;
                char title[SCE_SAMPLE_UTIL_SAVE_DATA_TITLE_MAXSIZE];
                char subTitle[SCE_SAMPLE_UTIL_SAVE_DATA_SUBTITLE_MAXSIZE];
                char detail[SCE_SAMPLE_UTIL_SAVE_DATA_DETAIL_MAXSIZE];
                uint32_t userParam;
                char iconPath[SCE_SAMPLE_UTIL_SAVE_DATA_SLOT_ICON_PATH_MAXSIZE];
                SceDateTime mtime;
            } SaveDataSlotParam;
        }
    }
}
```

Members

<i>status</i>	Save data slot status.
<i>title</i>	Title name (terminated with NULL, UTF-8)
<i>subTitle</i>	Sub-title name (terminated with NULL, UTF-8)
<i>detail</i>	Detailed information (terminated with NULL, UTF-8)
<i>userParam</i>	User parameter.
<i>iconPath</i>	Thumbnail icon path (terminated with NULL)
<i>mtime</i>	Last modified date/time.

Description

This is the save data slot parameter structure.

SCE CONFIDENTIAL

SaveDataSearchCond

Save data slot search condition structure.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            typedef
            struct sce::SampleUtil::System::SaveDataSearchCond {
                SaveDataSlotId from;
                uint32_t range;
                SaveDataSlotSortKey key;
                SaveDataSlotSortOrder order;
            } SaveDataSearchCond;
        }
    }
}
```

Members

<i>from</i>	Search start save data slot ID.
<i>range</i>	Search range.
<i>key</i>	Save data slot search sort key.
<i>order</i>	Save data slot search sort order.

Description

This is the save data slot search condition structure.

SCE CONFIDENTIAL

SaveDataSearchResult

Save data slot search result structure.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            typedef struct sce::SampleUtil::System::SaveDataSearchResult {
                uint32_t hitNum;
                SaveDataSlotId *slotList;
                uint32_t slotListNum;
            } SaveDataSearchResult;
        }
    }
}
```

Members

<i>hitNum</i>	Number of search hits.
<i>slotList</i>	Save data slot ID list.
<i>slotListNum</i>	Number of save data slot ID lists.

Description

This is the save data slot search result structure. For slotList, specify a pointer to the array to store the slot ID list for the search results. For slotListNum, specify the number of arrays specified for slotList.

DataSaveItem

Structure for specifying target save data to save.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            typedef struct sce::SampleUtil::System::DataSaveItem {
                const char *dataPath;
                const void *buf;
                size_t bufSize;
                uint32_t offset;
                SaveDataSaveMode mode;
            } DataSaveItem;
        }
    }
}
```

Members

<i>dataPath</i>	Data path of the target save data to save.
<i>buf</i>	Buffer where the target save data content to save is stored when saving a file.
<i>bufSize</i>	Size of the target save data to save when saving a file.
<i>offset</i>	Write offset position from the file start for the target save data to save when saving a file.
<i>mode</i>	Specify save mode for the target save data to save.

Description

This structure specifies the target save data to save. For *dataPath*, specify the data path of the target save data to save. With PlayStation®Vita, a path with a directory hierarchy up to three levels deep can be specified for *dataPath*. Relative paths using ".." are not valid. For *buf*, specify the address of the buffer where the content of the save data to save is stored. Accordingly, specify the size of the buffer for *bufSize*. For *offset*, specify the write offset position from the file start when saving the save data content specified with *buf*. When specifying *kSaveDataSaveModeDirectory* for *mode*, NULL must be set for *buf* and 0 must be set for both *bufSize* and *offset*.

SCE CONFIDENTIAL

SaveDataMountPoint

Save data mount point structure.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            typedef
            struct sce::SampleUtil::System::SaveDataMountPoint {
                SCE_SAMPLE_TCHAR
                data[SCE_SAMPLE_UTIL_SAVE_DATA_MOUNT_POINT_DATA_MAXSIZE];
            } SaveDataMountPoint;
        }
    }
}
```

Members

data Mount point name (terminated with NULL, UTF-8)

Description

This structure receives the save data mount point name.

SCE CONFIDENTIAL

DataRemoveItem

Structure for specifying target save data to remove.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            typedef struct sce::SampleUtil::System::DataRemoveItem {
                const char *dataPath;
            } DataRemoveItem;
        }
    }
}
```

Members

dataPath Data path of the target save data to remove.

Description

This structure specifies the target save data to remove. For *dataPath*, specify the data path of the target save data to remove.

SCE CONFIDENTIAL

SaveDataOption

Structure for initializing [SaveData](#).

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            typedef struct sce::SampleUtil::System::SaveDataOption {
                char titleId[SCE_SAMPLE_UTIL_SAVE_DATA_TITLE_ID_DATA_SIZE];
                char passCode[SCE_SAMPLE_UTIL_SAVE_DATA_PASSCODE_DATA_SIZE];
            } SaveDataOption;
        }
    }
}
```

Members

titleId Title ID.
passCode Passcode.

Description

This is the structure for initializing [SaveData](#). This is used by specifying it to the option argument of [createSaveData\(\)](#).

Functions

createUserIdManager

Initialization function of [UserIdManager](#).

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            int createUserIdManager(
                UserIdManager **outUserIdManager,
                UserIdManagerOption *option = NULL
            );
        }
    }
}
```

Arguments

<code>outUserIdManager</code> <code>option</code>	Pointer to be returned by the generated UserIdManager . This is the UserIdManagerOption structure.
--	---

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This is the initialization function of [UserIdManager](#). Call this first when using [UserIdManager](#).

SCE CONFIDENTIAL

createSaveData

Create save data class instance.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            int createSaveData(
                SaveData **outSaveData,
                const SaveDataOption *option
            );
        }
    }
}
```

Arguments

<i>outSaveData</i>	Pointer to which the generated save data class instance returns
<i>option</i>	SaveDataOption structure.

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This creates a save data class instance. To delete the generated instance, use [sce::SampleUtil::destroy\(\)](#). When NULL is specified for option, the save data directory that is automatically mounted when the application starts will be accessed. When a value other than NULL is specified for option, the titleId save data directory specified for the [SaveDataOption](#) structure will be mounted. However, with PlayStation®Vita it will be mounted with read access.

sce::SampleUtil::System::UserIdList

000004892117

Summary

sce::SampleUtil::System::UserIdList

Structure that stores the user ID list.

Definition

```
#include <system.h>
struct UserIdList {};
```

Description

This structure stores the user ID list.

Fields

Public Instance Fields

[UserId](#) *userId* This is the user ID list.

SCE CONFIDENTIAL

sce::SampleUtil::System::UserLoginStat usChangedEvents

Summary

sce::SampleUtil::System::UserLoginStatusChanged Events

Structure that stores the user login status (logged in/logged out) change events.

Definition

```
#include <system.h>
struct UserLoginStatusChangedEvents { };
```

Description

This structure stores the user login status (logged in/logged out) change events.

Fields

Public Instance Fields

[UserIdList](#) joinedUserIdList
[UserIdList](#) leftUserIdList

User ID list of logged in users that joined the game.

User ID list of users that have logged out and left the game.

SCE CONFIDENTIAL

sce::SampleUtil::System::UserIdManager

000004892117

Summary

sce::SampleUtil::System::UserIdManager

Class for managing user IDs.

Definition

```
#include <system.h>
class UserIdManager : public sce::SampleUtil::Resource { };
```

Description

[UserIdManager](#) is a class for supporting the management of user IDs.

Methods Summary

Methods	Description
~UserIdManager	Destructor.
update	Update.
getLoginUserIdList	Get user ID list of logged in users.
getUserLoginStatusChangedEventsOfLastUpdate	Get user login status (logged in/logged out) change event for last update.
getInitialUser	Obtains the user ID of the user that started the game.
getUserName	Obtains the user name.
getUserColor	Get user color.

Constructors and Destructors

~UserIdManager

Destructor.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            class UserIdManager {
                virtual inline ~UserIdManager(void);
            }
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Public Instance Methods

update

Update.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            class UserIdManager {
                virtual int update(void)=0;
            }
        }
    }
}
```

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This is the update function of [UserIdManager](#). The [UserIdManager](#) user ID list updates are performed in this function.

SCE CONFIDENTIAL

getLoginUserIdList

Get user ID list of logged in users.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            class UserIdManager {
                virtual int getLoginUserIdList(
                    UserIdList *list
                )=0;
            }
        }
    }
}
```

Arguments

list [UserIdList](#) pointer

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This gets the user ID list of the logged in users.

SCE CONFIDENTIAL

getUserLoginStatusChangedEventsOfLastUpdate

Get user login status (logged in/logged out) change event for last update.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            class UserIdManager {
                virtual int
                getUserLoginStatusChangedEventsOfLastUpdate(  

                    UserLoginStatusChangedEvents *events  

                    )=0;  

            }
        }
    }
}
```

Arguments

events [UserLoginStatusChangedEvents](#) pointer

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This gets the user login status (logged in/logged out) change event for the last update.

SCE CONFIDENTIAL

getInitialUser

Obtains the user ID of the user that started the game.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            class UserIdManager {
                virtual int getInitialUser(
                    UserId *userId
                )=0;
            }
        }
    }
}
```

Arguments

userId Pointer that stores user IDs

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

Obtains the user ID of the user that started the game. In PlayStation®4, this function cannot be used with applications that have the InitialUserAlwaysLoggedIn flag set to disabled in param.sfo, and SCE_USER_SERVICE_ERROR_OPERATION_NOT_SUPPORTED will return.

SCE CONFIDENTIAL

getUserName

Obtains the user name.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            class UserIdManager {
                virtual int getUserName(
                    const UserId userId,
                    char *userName,
                    const size_t size
                )=0;
            }
        }
    }
}
```

Arguments

<i>userId</i>	User ID
<i>userName</i>	Pointer for the array to store the user name
<i>size</i>	Size of userName

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

Gets the user name.

SCE CONFIDENTIAL

getUserColor

Get user color.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            class UserIdManager {
                virtual int getUserColor(
                    const UserId userId,
                    UserColor *userColor
                )=0;
            }
        }
    }
}
```

Arguments

<i>userId</i>	User ID
<i>userColor</i>	Pointer that stores user color

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This obtains the user color (color allocated by the system to a user on PlayStation®4). Regarding user colors on PlayStation®4, refer to "User Management Overview" document.

sce::SampleUtil::System::UserIdManagerOption

Summary

sce::SampleUtil::System::UserIdManagerOption

Options structure for [UserIdManager](#).

Definition

```
#include <system.h>
struct UserIdManagerOption {};
```

Description

This is an options structure for [UserIdManager](#). This is used by specifying it to the option argument of [createUserIdManager\(\)](#).

sce::SampleUtil::System::SaveDataSlot Param

Summary

sce::SampleUtil::System::SaveDataSlotParam

Save data slot parameter structure.

Definition

```
#include <system.h>
struct SaveDataSlotParam { };
```

Description

This is the save data slot parameter structure.

Fields

Public Instance Fields

SaveDataSlotStatus <i>status</i>	Save data slot status.
char <i>title</i>	Title name (terminated with NULL, UTF-8)
char <i>subTitle</i>	Sub-title name (terminated with NULL, UTF-8)
char <i>detail</i>	Detailed information (terminated with NULL, UTF-8)
uint32_t <i>userParam</i>	User parameter.
char <i>iconPath</i>	Thumbnail icon path (terminated with NULL)
SceDateTime <i>mtime</i>	Last modified date/time.

SCE CONFIDENTIAL

sce::SampleUtil::System::SaveDataSearchCond

Summary

sce::SampleUtil::System::SaveDataSearchCond

Save data slot search condition structure.

Definition

```
#include <system.h>
struct SaveDataSearchCond { };
```

Description

This is the save data slot search condition structure.

Fields

Public Instance Fields

[SaveDataSlotId](#) *from*

Search start save data slot ID.

[uint32_t](#) *range*

Search range.

[SaveDataSlotSortKey](#) *key*

Save data slot search sort key.

[SaveDataSlotSortOrder](#) *order*

Save data slot search sort order.

SCE CONFIDENTIAL

sce::SampleUtil::System::SaveDataSearchResult

Summary

sce::SampleUtil::System::SaveDataSearchResult

Save data slot search result structure.

Definition

```
#include <system.h>
struct SaveDataSearchResult { };
```

Description

This is the save data slot search result structure. For slotList, specify a pointer to the array to store the slot ID list for the search results. For slotListNum, specify the number of arrays specified for slotList.

Fields

Public Instance Fields

uint32_t <i>hitNum</i>	Number of search hits.
SaveDataSlotId * <i>slotList</i>	Save data slot ID list.
uint32_t <i>slotListNum</i>	Number of save data slot ID lists.

SCE CONFIDENTIAL

sce::SampleUtil::System::DataSaveItem

000004892117

Summary

sce::SampleUtil::System::DataSaveItem

Structure for specifying target save data to save.

Definition

```
#include <system.h>
struct DataSaveItem {};
```

Description

This structure specifies the target save data to save. For dataPath, specify the data path of the target save data to save. With PlayStation®Vita, a path with a directory hierarchy up to three levels deep can be specified for dataPath. Relative paths using ".." are not valid. For buf, specify the address of the buffer where the content of the save data to save is stored. Accordingly, specify the size of the buffer for bufSize. For offset, specify the write offset position from the file start when saving the save data content specified with buf. When specifying kSaveDataSaveModeDirectory for mode, NULL must be set for buf and 0 must be set for both bufSize and offset.

Fields

Public Instance Fields

const char * <i>dataPath</i>	Data path of the target save data to save.
const void * <i>buf</i>	Buffer where the target save data content to save is stored when saving a file.
size_t <i>bufSize</i>	Size of the target save data to save when saving a file.
uint32_t <i>offset</i>	Write offset position from the file start for the target save data to save when saving a file.
SaveDataSaveMode <i>mode</i>	Specify save mode for the target save data to save.

sce::SampleUtil::System::SaveDataMousePoint

Summary

sce::SampleUtil::System::SaveDataMountPoint

Save data mount point structure.

Definition

```
#include <system.h>
struct SaveDataMountPoint {};
```

Description

This structure receives the save data mount point name.

Fields

Public Instance Fields

SCE_SAMPLE_TCHAR <i>data</i>	Mount point name (terminated with NULL, UTF-8)
------------------------------	--

SCE CONFIDENTIAL

sce::SampleUtil::System::DataRemovelt em

000004892117

Summary

sce::SampleUtil::System::DataRemoveItem

Structure for specifying target save data to remove.

Definition

```
#include <system.h>
struct DataRemoveItem { };
```

Description

This structure specifies the target save data to remove. For *dataPath*, specify the data path of the target save data to remove.

Fields

Public Instance Fields

const char * <i>dataPath</i>	Data path of the target save data to remove.
------------------------------	--

SCE CONFIDENTIAL

sce::SampleUtil::System::SaveData

000004892117

Summary

sce::SampleUtil::System::SaveData

Save data.

Definition

```
#include <system.h>
class SaveData : public sce::SampleUtil::Resource {};
```

Description

This class provides the save data feature. On PlayStation®4, save data is created per user ID.

Methods Summary

Methods	Description
slotCreate	Create save data slot.
slotDelete	Delete save data slot.
slotSetParam	Set save data slot parameters.
slotGetParam	Get save data slot parameters.
slotSearch	Search for save data slot.
dataSave	Save save data.
atomicSlotCreateAndDataSave	Create save data slot and save save data atomically.
atomicDataSaveAndSlotSetParam	Save save data and set save data slot parameters atomically.
dataLoadMount	Mount save data directory for loading.
dataLoadUmount	Unmount save data directory.
dataRemove	Remove save data.

Public Instance Methods

slotCreate

Create save data slot.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            class SaveData {
                virtual int slotCreate(
                    const UserId userId,
                    const SaveDataSlotId slotId,
                    const SaveDataSlotParam *param,
                    const size_t sizeKiB
                )=0;
            }
        }
    }
}
```

Arguments

<i>userId</i>	User ID. This parameter is ignored on all platforms other than PlayStation®4.
<i>slotId</i>	Save data slot ID (0 to SCE_SAMPLE_UTIL_SAVE_DATA_SLOT_MAX-1)
<i>param</i>	Save data slot parameter structure
<i>sizeKiB</i>	Data size (KiB). Specify 10240 or more for PlayStation®4. Moreover, this value is aligned to a multiple of 32 on PlayStation®4.

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This creates a save data slot.

SCE CONFIDENTIAL

slotDelete

Delete save data slot.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            class SaveData {
                virtual int slotDelete(
                    const UserId userId,
                    const SaveDataSlotId slotId
                )=0;
            }
        }
    }
}
```

Arguments

<i>userId</i>	User ID. This parameter is ignored on all platforms other than PlayStation®4.
<i>slotId</i>	Save data slot ID (0 to SCE_SAMPLE_UTIL_SAVE_DATA_SLOT_MAX-1)

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This deletes a save data slot.

SCE CONFIDENTIAL

slotSetParam

Set save data slot parameters.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            class SaveData {
                virtual int slotSetParam(
                    const UserId userId,
                    const SaveDataSlotId slotId,
                    const SaveDataSlotParam *param
                )=0;
            }
        }
    }
}
```

Arguments

<i>userId</i>	User ID. This parameter is ignored on all platforms other than PlayStation®4.
<i>slotId</i>	Save data slot ID (0 to SCE_SAMPLE_UTIL_SAVE_DATA_SLOT_MAX-1)
<i>param</i>	Save data slot parameter structure

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This sets save data slot parameters. With PlayStation®4, when an error returns, some of the parameters may be set.

SCE CONFIDENTIAL

slotGetParam

Get save data slot parameters.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            class SaveData {
                virtual int slotGetParam(
                    const UserId userId,
                    const SaveDataSlotId slotId,
                    SaveDataSlotParam *outParam
                )=0;
            }
        }
    }
}
```

Arguments

<i>userId</i>	User ID. This parameter is ignored on all platforms other than PlayStation®4.
<i>slotId</i>	Save data slot ID (0 to SCE_SAMPLE_UTIL_SAVE_DATA_SLOT_MAX-1)
<i>outParam</i>	Save data slot parameter structure

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This gets save data slot parameters.

SCE CONFIDENTIAL

slotSearch

Search for save data slot.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            class SaveData {
                virtual int slotSearch(
                    const UserId userId,
                    const SaveDataSearchCond *cond,
                    SaveDataSearchResult *outResult
                )=0;
            }
        }
    }
}
```

Arguments

<i>userId</i>	User ID. This parameter is ignored on all platforms other than PlayStation®4.
<i>cond</i>	Save data slot search condition structure
<i>outResult</i>	Save data slot search result structure

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This searches for a save data slot. For the search processing results, the number of matching save data slot hits will be stored in the hitNum member of outResult, and a save data slot ID array will be stored in the slotList member of the argument outResult.

dataSave

Save save data.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            class SaveData {
                virtual int dataSave(
                    const UserId userId,
                    const SaveDataSlotId slotId,
                    const DataSaveItem *data,
                    const uint32_t dataNum,
                    size_t *outRequiredSizeKiB = NULL
                )=0;
            }
        }
    }
}
```

Arguments

<i>userId</i>	User ID. This parameter is ignored on all platforms other than PlayStation®4.
<i>slotId</i>	Save data slot ID (0 to SCE_SAMPLE_UTIL_SAVE_DATA_SLOT_MAX-1)
<i>data</i>	<i>DataSaveItem</i> array pointer. Specify target data to save in file or directory units. The upper limit for total data file size on PlayStation®Vita is 1 MiB.
<i>dataNum</i>	Number of target data to save. For PlayStation®Vita, 1 to SCE_APPUTIL_SAVEDATA_DATA_MAXNUM can be specified.
<i>outRequiredSizeKiB</i>	Size lacking for saving the save data specified in data (units in KiB) (when an SCE_APPUTIL_ERROR_SAVEDATA_NO_SPACE_QUOTA error or SCE_APPUTIL_ERROR_SAVEDATA_NO_SPACE_FS error occurs on PlayStation®Vita)

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This function saves save data specified in data in file or directory units. With PlayStation®Vita, 0 is stored in *outRequiredSizeKiB* and *outRequiredSizeKiB* returns for normal termination of the function. If an SCE_APPUTIL_ERROR_SAVEDATA_NO_SPACE_QUOTA error or SCE_APPUTIL_ERROR_SAVEDATA_NO_SPACE_FS error occurs, the size (KiB) insufficient for saving the save data specified for data and *dataNum* will be stored in *outRequiredSizeKiB* and *outRequiredSizeKiB* will return. With PlayStation®4, the value will not be stored in *outRequiredSizeKiB*.

SCE CONFIDENTIAL

atomicSlotCreateAndDataSave

Create save data slot and save save data atomically.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            class SaveData {
                virtual int atomicSlotCreateAndDataSave(
                    const SaveDataSlotId slotId,
                    const DataSaveItem *data,
                    const uint32_t dataNum,
                    const SaveDataSlotParam *slotParam,
                    const size_t slotSizeKiB
                )=0;
            }
        }
    }
}
```

Arguments

<i>slotId</i>	Save data slot ID (0 to SCE_SAMPLE_UTIL_SAVE_DATA_SLOT_MAX-1)
<i>data</i>	<i>DataSaveItem</i> array pointer. Specify target data to save in file or directory units. The upper limit is 1 MiB for the data's total file size.
<i>dataNum</i>	Number of target data to save. 1 to SCE_APPUTIL_SAVEDATA_DATA_MAXNUM can be specified.
<i>slotParam</i>	Save data slot parameter structure. An error will return if NULL is specified.
<i>slotSizeKiB</i>	Data size of the save data slot (KiB).

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This function creates a save data slot based on the settings specified by slotParam and atomically saves save data specified in data in file or directory units. An error will return if the save data slot which has the ID specified for slotId already exists.

SCE CONFIDENTIAL

atomicDataSaveAndSlotSetParam

Save save data and set save data slot parameters atomically.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            class SaveData {
                virtual int atomicDataSaveAndSlotSetParam(
                    const SaveDataSlotId slotId,
                    const DataSaveItem *data,
                    const uint32_t dataNum,
                    const SaveDataSlotParam *slotParam
                )=0;
            }
        }
    }
}
```

Arguments

<i>slotId</i>	Save data slot ID (0 to SCE_SAMPLE_UTIL_SAVE_DATA_SLOT_MAX-1)
<i>data</i>	<i>DataSaveItem</i> array pointer. Specify target data to save in file or directory units. The upper limit is 1 MiB for the data's total file size.
<i>dataNum</i>	Number of target data to save. 1 to SCE_APPUTIL_SAVEDATA_DATA_MAXNUM can be specified.
<i>slotParam</i>	Save data slot parameter structure. An error will return if NULL is specified.

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This saves the save data specified for data at the file or directory level and sets save data slot parameters specified for slotParam atomically. An error will return if the save data slot which has the ID specified for slotId does not exist.

SCE CONFIDENTIAL

dataLoadMount

Mount save data directory for loading.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            class SaveData {
                virtual int dataLoadMount(
                    const UserId userId,
                    const SaveDataSlotId slotId,
                    SaveDataMountPoint *outMountPoint
                )=0;
            }
        }
    }
}
```

Arguments

<i>userId</i>	User ID. This parameter is ignored on all platforms other than PlayStation®4.
<i>slotId</i>	Save data slot ID (0 to SCE_SAMPLE_UTIL_SAVE_DATA_SLOT_MAX-1)
<i>outMountPoint</i>	Storage destination for the allocated mount point name

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This mounts a save data slot directory for loading. Use an obtained mount point name to load the save data. With PlayStation®Vita, only the mount point name for loading will be returned; actual mount processing will not be performed in this function.

SCE CONFIDENTIAL

dataLoadUnmount

Unmount save data directory.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            class SaveData {
                virtual int dataLoadUnmount(
                    const SaveDataMountPoint *mountPoint
                )=0;
            }
        }
    }
}
```

Arguments

mountPoint Mount point name

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This unmounts a save data slot directory mounted for loading. With PlayStation®Vita, actual unmount processing is not performed in this function.

SCE CONFIDENTIAL

dataRemove

Remove save data.

Definition

```
#include <system.h>
namespace sce {
    namespace SampleUtil {
        namespace System {
            class SaveData {
                virtual int dataRemove(
                    const UserId userId,
                    const SaveDataSlotId slotId,
                    const DataRemoveItem *data,
                    const uint32_t dataNum
                )=0;
            }
        }
    }
}
```

Arguments

<i>userId</i>	User ID. This parameter is ignored on all platforms other than PlayStation®4.
<i>slotId</i>	Save data slot ID (0 to SCE_SAMPLE_UTIL_SAVE_DATA_SLOT_MAX-1)
<i>data</i>	DataRemoveItem array pointer (specify save data (file or directory level) to remove)
<i>dataNum</i>	Number of save data to remove. For PlayStation®Vita, 1 to SCE_APPUTIL_SAVEDATA_DATA_MAXNUM can be specified.

Return Values

Value	Description
SCE_OK	Success
(<0)	Error code

Description

This removes the save data specified for data and dataNum at the file or directory level. This removes the file/directory only without deleting the save data slot.

**sce::SampleUtil::System::SaveDataOpti
on**

Summary

sce::SampleUtil::System::SaveDataOption

Structure for initializing [SaveData](#).

Definition

```
#include <system.h>
struct SaveDataOption { };
```

Description

This is the structure for initializing [SaveData](#). This is used by specifying it to the option argument of [createSaveData\(\)](#).

Fields

Public Instance Fields

char *titleId* Title ID.
char *passCode* Passcode.