

YAGE

1.0

Generated by Doxygen 1.8.11



# Contents

<b>1</b>	<b>YAGE (Yet Another Game Engine)</b>	<b>1</b>
<b>2</b>	<b>Hierarchical Index</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class List . . . . .	5
<b>4</b>	<b>Class Documentation</b>	<b>7</b>
4.1	AmbientLight Class Reference . . . . .	7
4.1.1	Detailed Description . . . . .	7
4.1.2	Constructor & Destructor Documentation . . . . .	7
4.1.2.1	AmbientLight(glm::vec3 color, float strength=1.0) . . . . .	7
4.1.3	Member Function Documentation . . . . .	7
4.1.3.1	Draw(GLuint) override final . . . . .	7
4.2	Camera Class Reference . . . . .	8
4.2.1	Detailed Description . . . . .	8
4.2.2	Constructor & Destructor Documentation . . . . .	9
4.2.2.1	Camera(Camera const &)=delete . . . . .	9
4.2.3	Member Function Documentation . . . . .	9
4.2.3.1	ComputeMatrices() . . . . .	9
4.2.3.2	GetEyeDirection() . . . . .	9
4.2.3.3	GetInstance() . . . . .	9
4.2.3.4	GetProjectionMatrix() . . . . .	9
4.2.3.5	GetViewMatrix() . . . . .	9

4.2.3.6	operator=(Camera const &)=delete . . . . .	9
4.2.3.7	resizeWindow(float width, float height) . . . . .	9
4.2.4	Member Data Documentation . . . . .	9
4.2.4.1	aspect . . . . .	9
4.2.4.2	fov . . . . .	10
4.2.4.3	zFar . . . . .	10
4.2.4.4	zNear . . . . .	10
4.3	ContextInfo Struct Reference . . . . .	10
4.3.1	Constructor & Destructor Documentation . . . . .	10
4.3.1.1	ContextInfo() . . . . .	10
4.3.1.2	ContextInfo(int major_version, int minor_version, bool core) . . . . .	10
4.3.2	Member Function Documentation . . . . .	10
4.3.2.1	operator=(const ContextInfo &info) . . . . .	10
4.3.3	Member Data Documentation . . . . .	10
4.3.3.1	core . . . . .	10
4.3.3.2	major_version . . . . .	10
4.3.3.3	minor_version . . . . .	10
4.4	DebugDrawer Class Reference . . . . .	10
4.4.1	Constructor & Destructor Documentation . . . . .	11
4.4.1.1	DebugDrawer() . . . . .	11
4.4.1.2	~DebugDrawer() . . . . .	11
4.4.2	Member Function Documentation . . . . .	11
4.4.2.1	clearBuffers() . . . . .	11
4.4.2.2	draw3dText(const btVector3 &location, const char *textString) override . . . . .	11
4.4.2.3	drawContactPoint(const btVector3 &PointOnB, const btVector3 &normalOnB, bt↔ Scalar distance, int lifeTime, const btVector3 &color) override . . . . .	11
4.4.2.4	drawLine(const btVector3 &from, const btVector3 &to, const btVector3 &color) override . . . . .	11
4.4.2.5	getDebugMode() const override . . . . .	11
4.4.2.6	render() . . . . .	11
4.4.2.7	reportErrorWarning(const char *warningString) override . . . . .	11

4.4.2.8	setDebugMode(int debugMode) override . . . . .	11
4.5	DirectionalLight Class Reference . . . . .	11
4.5.1	Detailed Description . . . . .	12
4.5.2	Constructor & Destructor Documentation . . . . .	12
4.5.2.1	DirectionalLight(glm::vec3 color, glm::vec3 position, glm::vec3 halfVector) . . . . .	12
4.5.3	Member Function Documentation . . . . .	12
4.5.3.1	Draw(GLuint) override final . . . . .	12
4.5.3.2	DrawShadow(GLuint) override final . . . . .	12
4.6	DVertexFormat Struct Reference . . . . .	12
4.6.1	Constructor & Destructor Documentation . . . . .	13
4.6.1.1	DVertexFormat(const glm::vec3 &inPos, const glm::vec3 &inColor) . . . . .	13
4.6.2	Member Data Documentation . . . . .	13
4.6.2.1	color . . . . .	13
4.6.2.2	position . . . . .	13
4.7	FramebufferInfo Struct Reference . . . . .	13
4.7.1	Constructor & Destructor Documentation . . . . .	13
4.7.1.1	FramebufferInfo() . . . . .	13
4.7.1.2	FramebufferInfo(bool color, bool depth, bool stencil, bool msaa) . . . . .	13
4.7.2	Member Data Documentation . . . . .	13
4.7.2.1	flags . . . . .	13
4.7.2.2	msaa . . . . .	13
4.8	GameObjectsBuilder Class Reference . . . . .	13
4.8.1	Constructor & Destructor Documentation . . . . .	14
4.8.1.1	GameObjectsBuilder() . . . . .	14
4.8.1.2	~GameObjectsBuilder() . . . . .	14
4.8.2	Member Function Documentation . . . . .	14
4.8.2.1	addLight(GameObjectType type) . . . . .	14
4.8.2.2	addModel(const std::string modelPath, const std::string texturePath=std::string()) . . . . .	14
4.8.2.3	addParticleSystem(const std::string &texturePath) . . . . .	14
4.8.2.4	addRigidBody(float mass) . . . . .	14

4.8.2.5	copyModel()	14
4.8.2.6	getResult() const	14
4.8.2.7	lockUpright()	15
4.8.2.8	setAttenuation(float constant, float linear, float quadratic)	15
4.8.2.9	setCastsShadows(bool)	15
4.8.2.10	setColor(glm::vec3 color)	15
4.8.2.11	setConeDirection(glm::vec3 coneDirection)	15
4.8.2.12	setDiffuse(const std::string &)	15
4.8.2.13	setHalfVector(glm::vec3 halfVector)	15
4.8.2.14	setNormal(const std::string &)	15
4.8.2.15	setParticleCount(int)	15
4.8.2.16	setParticleLife(float)	15
4.8.2.17	setParticleSpawnRate(int)	15
4.8.2.18	setPosition(glm::vec3)	15
4.8.2.19	setRotation(float angleX, float angleY, float angleZ)	15
4.8.2.20	setScale(float)	15
4.8.2.21	setSpecular(const std::string &)	15
4.8.2.22	setSpotCutoff(float cutoff)	15
4.8.2.23	setSpotExponent(float exponent)	15
4.8.2.24	setStrength(float strength)	15
4.9	IGameObject Class Reference	15
4.9.1	Detailed Description	16
4.9.2	Constructor & Destructor Documentation	16
4.9.2.1	~IGameObject()	16
4.9.3	Member Function Documentation	16
4.9.3.1	Destroy()=0	16
4.9.3.2	Draw(GLuint shader)=0	16
4.9.3.3	DrawShadow(GLuint shader)=0	17
4.9.3.4	GetProgram()	17
4.9.3.5	GetShadowProgram()	17

4.9.3.6	GetTexture(const std::string &textureName) const =0 . . . . .	17
4.9.3.7	SetProgram(GLuint shaderName) . . . . .	18
4.9.3.8	SetShadowProgram(GLuint shaderName) . . . . .	18
4.9.3.9	SetTexture(const std::string &textureName, const TextureType &textureType, const GLuint &texture)=0 . . . . .	18
4.9.3.10	Update()=0 . . . . .	18
4.9.4	Member Data Documentation . . . . .	19
4.9.4.1	program . . . . .	19
4.9.4.2	shadowProgram . . . . .	19
4.10	Init_GLEW Class Reference . . . . .	19
4.10.1	Detailed Description . . . . .	19
4.10.2	Constructor & Destructor Documentation . . . . .	19
4.10.2.1	Init_GLEW() . . . . .	19
4.10.2.2	~Init_GLEW() . . . . .	19
4.10.3	Member Function Documentation . . . . .	19
4.10.3.1	Init() . . . . .	19
4.11	Init_GLUT Class Reference . . . . .	19
4.11.1	Detailed Description . . . . .	20
4.11.2	Member Function Documentation . . . . .	20
4.11.2.1	close() . . . . .	20
4.11.2.2	enterFullscreen() . . . . .	20
4.11.2.3	exitFullscreen() . . . . .	20
4.11.2.4	init(const WindowInfo &, const ContextInfo &, const FrameBufferInfo &) . . . . .	20
4.11.2.5	printOpenGLInfo(const WindowInfo &window, const ContextInfo &context) . . . . .	20
4.11.2.6	run() . . . . .	20
4.11.2.7	setListener(Scene_Manager * &iListener) . . . . .	20
4.12	Light Class Reference . . . . .	20
4.12.1	Detailed Description . . . . .	22
4.12.2	Constructor & Destructor Documentation . . . . .	22
4.12.2.1	Light() . . . . .	22
4.12.2.2	~Light() . . . . .	22

4.12.3	Member Function Documentation . . . . .	22
4.12.3.1	Destroy() override . . . . .	22
4.12.3.2	DisableShadows() . . . . .	22
4.12.3.3	Draw(GLuint) override . . . . .	22
4.12.3.4	DrawShadow(GLuint) override . . . . .	23
4.12.3.5	EnableShadows() . . . . .	23
4.12.3.6	GetTexture(const std::string &) const override . . . . .	23
4.12.3.7	SetAttenuation(float constant=0.0f, float linear=1.0f, float quadratic=0.0f) . . . . .	23
4.12.3.8	SetTexture(const std::string &, const TextureType &, const GLuint &) override . . . . .	24
4.12.3.9	Update() override . . . . .	24
4.12.4	Member Data Documentation . . . . .	24
4.12.4.1	_id . . . . .	24
4.12.4.2	ambient . . . . .	24
4.12.4.3	castsShadow . . . . .	24
4.12.4.4	color . . . . .	24
4.12.4.5	coneDirection . . . . .	24
4.12.4.6	constantAttenuation . . . . .	25
4.12.4.7	halfVector . . . . .	25
4.12.4.8	ids . . . . .	25
4.12.4.9	isEnabled . . . . .	25
4.12.4.10	linearAttenuation . . . . .	25
4.12.4.11	position . . . . .	25
4.12.4.12	quadraticAttenuation . . . . .	25
4.12.4.13	spotCosCutoff . . . . .	25
4.12.4.14	spotExponent . . . . .	25
4.12.4.15	texture . . . . .	25
4.12.4.16	type . . . . .	25
4.13	Light::LightUniformLocations Struct Reference . . . . .	25
4.13.1	Detailed Description . . . . .	26
4.13.2	Member Data Documentation . . . . .	26



4.13.2.1	ambient	26
4.13.2.2	color	26
4.13.2.3	coneDirection	26
4.13.2.4	constantAttenuation	26
4.13.2.5	halfVector	26
4.13.2.6	isEnabled	26
4.13.2.7	linearAttenuation	26
4.13.2.8	position	26
4.13.2.9	quadraticAttenuation	26
4.13.2.10	spotCosCutoff	26
4.13.2.11	spotExponent	26
4.13.2.12	type	26
4.14	Material Struct Reference	26
4.14.1	Detailed Description	27
4.15	Mesh Class Reference	27
4.15.1	Constructor & Destructor Documentation	28
4.15.1.1	Mesh(const aiMesh *, const aiMaterial *, Transform *)	28
4.15.1.2	Mesh(const Mesh *, Transform *)	28
4.15.1.3	~Mesh()	28
4.15.2	Member Function Documentation	28
4.15.2.1	Create()	28
4.15.2.2	Draw(GLuint) override final	28
4.15.2.3	DrawShadow(GLuint) override final	29
4.15.2.4	GetPositionVertices()	29
4.15.2.5	GetVertices() override final	29
4.15.2.6	Update() override final	29
4.15.3	Member Data Documentation	29
4.15.3.1	shininess	29
4.15.3.2	strength	29
4.16	Model Class Reference	29

4.16.1	Constructor & Destructor Documentation . . . . .	30
4.16.1.1	Model() . . . . .	30
4.16.1.2	Model(Transform *t) . . . . .	30
4.16.1.3	~Model() . . . . .	30
4.16.2	Member Function Documentation . . . . .	30
4.16.2.1	Destroy() override . . . . .	30
4.16.2.2	GetTexture(const std::string &) const override . . . . .	30
4.16.2.3	GetVao() const . . . . .	31
4.16.2.4	GetVbos() const . . . . .	31
4.16.2.5	GetVertices()=0 . . . . .	31
4.16.2.6	SetProgram(GLuint) override . . . . .	31
4.16.2.7	SetShadowProgram(GLuint) override . . . . .	31
4.16.2.8	SetTexture(const std::string &textureName, const TextureType &textureType, const GLuint &texture) override . . . . .	32
4.16.3	Member Data Documentation . . . . .	32
4.16.3.1	textures . . . . .	32
4.16.3.2	transform . . . . .	32
4.16.3.3	vao . . . . .	32
4.16.3.4	vbos . . . . .	32
4.17	Models_Manager Class Reference . . . . .	32
4.17.1	Detailed Description . . . . .	33
4.17.2	Constructor & Destructor Documentation . . . . .	33
4.17.2.1	Models_Manager() . . . . .	33
4.17.2.2	~Models_Manager() . . . . .	33
4.17.3	Member Function Documentation . . . . .	33
4.17.3.1	addLight(Light *light) . . . . .	33
4.17.3.2	CreateModel(const std::string &modelPath, const Transform &transform, const std::string &texturePath="", const TextureType type=Texture_Diffuse) . . . . .	33
4.17.3.3	CreateModel(const Scene_Container *&otherModel, const Transform &transform) . . . . .	33
4.17.3.4	CreateParticleSystem(Transform transform, const std::string &texturePath) . . . . .	34
4.17.3.5	Draw() . . . . .	34

4.17.3.6	DrawShadows()	34
4.17.3.7	Update()	34
4.18	Particle_Container Class Reference	34
4.18.1	Constructor & Destructor Documentation	35
4.18.1.1	Particle_Container(Transform t, const std::string &texturePath, bool enable=true, int max_particles=10000, int particle_rate=1000, float particle_max_life=5)	35
4.18.1.2	~Particle_Container()	35
4.18.2	Member Function Documentation	35
4.18.2.1	Destroy() override	35
4.18.2.2	Draw()	35
4.18.2.3	Draw(GLuint) override	35
4.18.2.4	DrawShadow()	36
4.18.2.5	DrawShadow(GLuint) override	36
4.18.2.6	GetTexture(const std::string &) const override	36
4.18.2.7	setMaxLife(float)	36
4.18.2.8	setMaxParticles(int)	36
4.18.2.9	setSpawnRate(int)	36
4.18.2.10	SetTexture(const std::string &, const TextureType &, const GLuint &) override	36
4.18.2.11	Update() override	37
4.18.3	Member Data Documentation	37
4.18.3.1	transform	37
4.19	Physics_Manager Class Reference	37
4.19.1	Constructor & Destructor Documentation	37
4.19.1.1	~Physics_Manager()	37
4.19.2	Member Function Documentation	37
4.19.2.1	AddConstraint(btTypedConstraint *)	37
4.19.2.2	AddRigidBody(btRigidBody *)	37
4.19.2.3	DrawDebug()	37
4.19.2.4	GetInstance()	37
4.19.2.5	Step()	37
4.20	PointLight Class Reference	38

4.20.1 Detailed Description . . . . .	38
4.20.2 Constructor & Destructor Documentation . . . . .	38
4.20.2.1 PointLight(glm::vec3 color, glm::vec3 position, float constantAttenuation=0.0f, float linearAttenuation=1.0f, float quadraticAttenuation=0.0f) . . . . .	38
4.20.3 Member Function Documentation . . . . .	38
4.20.3.1 Draw(GLuint) override final . . . . .	38
4.21 Scene_Container Class Reference . . . . .	39
4.21.1 Constructor & Destructor Documentation . . . . .	40
4.21.1.1 Scene_Container(const std::string &, Transform) . . . . .	40
4.21.1.2 Scene_Container(const Scene_Container *amp;, Transform) . . . . .	40
4.21.1.3 ~Scene_Container() . . . . .	40
4.21.2 Member Function Documentation . . . . .	40
4.21.2.1 Destroy() override . . . . .	40
4.21.2.2 Draw() . . . . .	40
4.21.2.3 Draw(GLuint) override . . . . .	40
4.21.2.4 DrawShadow() . . . . .	40
4.21.2.5 DrawShadow(GLuint) override . . . . .	40
4.21.2.6 GetMeshes() const . . . . .	40
4.21.2.7 getRigidBody() . . . . .	40
4.21.2.8 GetTexture(const std::string &) const override . . . . .	40
4.21.2.9 InitRigidBody(btScalar mass) . . . . .	41
4.21.2.10 SetProgram(GLuint) override . . . . .	41
4.21.2.11 SetShadowProgram(GLuint) override . . . . .	41
4.21.2.12 SetTexture(const std::string &, const TextureType &, const GLuint &) override . . . . .	42
4.21.2.13 Update() override . . . . .	42
4.21.3 Member Data Documentation . . . . .	42
4.21.3.1 transform . . . . .	42
4.22 Scene_Manager Class Reference . . . . .	42
4.22.1 Constructor & Destructor Documentation . . . . .	43
4.22.1.1 Scene_Manager(std::string) . . . . .	43
4.22.1.2 ~Scene_Manager() . . . . .	43

4.22.2	Member Function Documentation . . . . .	43
4.22.2.1	GetDeltaTime() . . . . .	43
4.22.2.2	GetFPS() . . . . .	43
4.22.2.3	notifyBeginFrame() . . . . .	43
4.22.2.4	notifyDisplayFrame() . . . . .	43
4.22.2.5	notifyEndFrame() . . . . .	43
4.22.2.6	notifyReshape(int width, int height, int p_width, int p_height) . . . . .	43
4.22.2.7	RenderPass() const . . . . .	43
4.22.2.8	SetupScene(const GameObjectsBuilder &) . . . . .	43
4.22.2.9	ShadowPass() const . . . . .	43
4.22.2.10	UpdatePass() const . . . . .	43
4.23	Shader_Factory Class Reference . . . . .	43
4.23.1	Constructor & Destructor Documentation . . . . .	44
4.23.1.1	~Shader_Factory(void) . . . . .	44
4.23.2	Member Function Documentation . . . . .	44
4.23.2.1	CreateDebugProgram() . . . . .	44
4.23.2.2	CreateProgram(const std::string &shaderName, const std::string &Vertex↵ ShaderFilename, const std::string &FragmentShaderFilename) . . . . .	44
4.23.2.3	GetInstance() . . . . .	44
4.23.2.4	SetTextureShader(IGameObject &model) . . . . .	44
4.24	Shadow_Manager Class Reference . . . . .	44
4.24.1	Detailed Description . . . . .	45
4.24.2	Member Function Documentation . . . . .	45
4.24.2.1	BindForWriting() const . . . . .	45
4.24.2.2	GetDepthMatrix() const . . . . .	45
4.24.2.3	GetInstance() . . . . .	45
4.24.2.4	GetShadowMap() const . . . . .	45
4.24.2.5	SetDepthMatrix(glm::mat4 DepthMatrix) . . . . .	45
4.24.2.6	Unbind() const . . . . .	46
4.24.3	Member Data Documentation . . . . .	46
4.24.3.1	DEPTH_TEXTURE_SIZE . . . . .	46

4.25 SpotLight Class Reference . . . . .	46
4.25.1 Detailed Description . . . . .	46
4.25.2 Constructor & Destructor Documentation . . . . .	47
4.25.2.1 SpotLight(glm::vec3 color, glm::vec3 position, glm::vec3 coneDirection, float spotCosCutoff=0.99f, float spotExponent=0.0f, float constantAttenuation=0.0f, float linearAttenuation=1.0f, float quadraticAttenuation=0.0f) . . . . .	47
4.25.3 Member Function Documentation . . . . .	47
4.25.3.1 Draw(GLuint) override final . . . . .	47
4.26 Texture Struct Reference . . . . .	47
4.26.1 Detailed Description . . . . .	47
4.26.2 Member Data Documentation . . . . .	47
4.26.2.1 id . . . . .	47
4.26.2.2 name . . . . .	47
4.26.2.3 type . . . . .	47
4.27 Transform Class Reference . . . . .	47
4.27.1 Detailed Description . . . . .	48
4.27.2 Constructor & Destructor Documentation . . . . .	49
4.27.2.1 Transform() . . . . .	49
4.27.2.2 Transform(glm::vec3 position, float scale, glm::quat rotation) . . . . .	49
4.27.2.3 ~Transform() . . . . .	49
4.27.3 Member Function Documentation . . . . .	49
4.27.3.1 getRotationMatrix() const . . . . .	49
4.27.3.2 getScaleMatrix() const . . . . .	49
4.27.3.3 getTransformMatrix() const . . . . .	49
4.27.3.4 getTranslationMatrix() const . . . . .	49
4.27.3.5 IncRotation(float angle, glm::vec3 axis) . . . . .	49
4.27.3.6 operator=(const btTransform &trans) . . . . .	50
4.27.3.7 RotateX(float angle) . . . . .	50
4.27.3.8 RotateY(float angle) . . . . .	50
4.27.3.9 RotateZ(float angle) . . . . .	50
4.27.3.10 SetPosition(const glm::vec3 &position) . . . . .	51

4.27.3.11	SetRotation(float angle, glm::vec3 axis) . . . . .	51
4.27.3.12	SetScale(float scale) . . . . .	51
4.27.4	Member Data Documentation . . . . .	51
4.27.4.1	position . . . . .	51
4.27.4.2	rotation . . . . .	51
4.27.4.3	scale . . . . .	51
4.28	VboIndexer Class Reference . . . . .	51
4.28.1	Member Function Documentation . . . . .	52
4.28.1.1	indexVBO(std::vector< glm::vec3 > &in_vertices, std::vector< glm::vec3 > &in_↵ _colors, std::vector< unsigned short > &out_indices, std::vector< DVertex↵ Format > &out_formats) . . . . .	52
4.29	VertexFormat Struct Reference . . . . .	52
4.29.1	Detailed Description . . . . .	52
4.29.2	Constructor & Destructor Documentation . . . . .	52
4.29.2.1	VertexFormat(const glm::vec3 &inPos, const glm::vec2 &inUV, const glm::vec3 &inNormal, const glm::vec3 &inTangent, const glm::vec3 &inBitangent) . . . . .	52
4.29.3	Member Function Documentation . . . . .	52
4.29.3.1	operator<(const VertexFormat that) const . . . . .	52
4.29.4	Friends And Related Function Documentation . . . . .	52
4.29.4.1	operator<< . . . . .	53
4.29.5	Member Data Documentation . . . . .	53
4.29.5.1	bitangent . . . . .	53
4.29.5.2	normal . . . . .	53
4.29.5.3	position . . . . .	53
4.29.5.4	tangent . . . . .	53
4.29.5.5	uv . . . . .	53
4.30	WindowInfo Struct Reference . . . . .	53
4.30.1	Detailed Description . . . . .	53
4.30.2	Constructor & Destructor Documentation . . . . .	54
4.30.2.1	WindowInfo() . . . . .	54
4.30.2.2	WindowInfo(std::string name, int position_x, int position_y, int width, int height, bool isReshapable) . . . . .	54
4.30.2.3	WindowInfo(const WindowInfo &windowInfo) . . . . .	54
4.30.3	Member Function Documentation . . . . .	54
4.30.3.1	operator=(const WindowInfo &windowInfo) . . . . .	54
4.30.4	Member Data Documentation . . . . .	54
4.30.4.1	height . . . . .	54
4.30.4.2	isReshapable . . . . .	54
4.30.4.3	name . . . . .	54
4.30.4.4	position_x . . . . .	54
4.30.4.5	position_y . . . . .	54
4.30.4.6	width . . . . .	54





# Chapter 1

## YAGE (Yet Another Game Engine)

This is a C++ based 3D game engine targeted for Windows.

### Requirements

- Microsoft Visual Studio 2015 (sorry, will port to gcc later)

### Getting Started

- Clone the repository
- Open `YAGE.sln` file with visual studio
- Install missing dependencies:
  - Tools > Nuget Package Manager > Package Manager Console
  - Type this into the the console: `Update-Package -Reinstall`
- Edit `main.cpp` and play with the game code
- Run the default startup project `F10`

### Documentation

If you are viewing this on Github, you can find our documentation here: <http://harrygogonis.github.io/YAGE>

### Screenshots

### Viewport Controls

Control	Action
W	Moves the camera forward.
S	Moves the camera backward.
A	Moves the camera left.
D	Moves the camera right.
E	Moves the camera up.
Q	Moves the camera down.
Mouse Scroll DOWN	Zooms the camera out (raises FOV).
Mouse Scroll UP	Zooms the camera in (lowers FOV).

### 3rd Party Libraries

- ASSIMP
- SOIL
- Bullet
- Freeglut
- OpenGL
- GLM

### Features

- Lighting system (Ambient, Point, Directional, Spotlight)
- Physics system (using bullet physics)
- Diffuse mapping
- Normal mapping
- Specular mapping
- Import scenes via .fbx, .obj, and more
- Game object builder
- Dynamic shadows
- Particle system

### Limitations

- Significant frame drop after ~500k polygons
- Shadows only supported w/ one directional light

### Future Features

- Animation
- Text rendering

### Helpful resources

- [in2gpu OpenGL Tutorial](#)
- [OpenGL Documentation](#)
- [Red Book](#)
- [OpenGL-Tutorial](#)
- [OGLDev](#)

### Authors

- Harry Gogonis
- Dylan Richardson

## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

btlDebugDraw	
DebugDrawer . . . . .	10
Camera . . . . .	8
ContextInfo . . . . .	10
DVertexFormat . . . . .	12
FrameBufferInfo . . . . .	13
GameObjectsBuilder . . . . .	13
IGameObject . . . . .	15
Light . . . . .	20
AmbientLight . . . . .	7
DirectionalLight . . . . .	11
PointLight . . . . .	38
SpotLight . . . . .	46
Model . . . . .	29
Mesh . . . . .	27
Particle_Container . . . . .	34
Scene_Container . . . . .	39
Init_GLEW . . . . .	19
Init_GLUT . . . . .	19
Light::LightUniformLocations . . . . .	25
Material . . . . .	26
Models_Manager . . . . .	32
Physics_Manager . . . . .	37
Scene_Manager . . . . .	42
Shader_Factory . . . . .	43
Shadow_Manager . . . . .	44
Texture . . . . .	47
Transform . . . . .	47
VboIndexer . . . . .	51
VertexFormat . . . . .	52
WindowInfo . . . . .	53



## Chapter 3

# Class Index

### 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<b>AmbientLight</b>	
An ambient light. Uniformly lights up everything . . . . .	7
<b>Camera</b>	
The game camera implemented as a Singleton. Primary use is to generate Model-View↔	
Projection matrices . . . . .	8
<b>ContextInfo</b> . . . . .	10
<b>DebugDrawer</b> . . . . .	10
<b>DirectionalLight</b>	
A directional light. Comes from "infinity" and lights up anything in the direction it is coming from	11
<b>DVertexFormat</b> . . . . .	12
<b>FramebufferInfo</b> . . . . .	13
<b>GameObjectsBuilder</b> . . . . .	13
<b>IGameObject</b>	
A Game Object. All objects that are to be rendered will inherit this class . . . . .	15
<b>Init_GLEW</b>	
Helper class to initialize GLEW . . . . .	19
<b>Init_GLUT</b>	
Helper class to initialize GLUT and setup listener callbacks . . . . .	19
<b>Light</b>	
A light is a special class of Game Object that mostly just sends uniform data to the shaders.	
Lights are fundamental to the shading pipeline! Without a light, nothing will show up in the scene	20
<b>Light::LightUniformLocations</b>	
Holds the various uniform location ID's of the <b>Light</b> (p. 20) . . . . .	25
<b>Material</b>	
A material. XXX Not yet implemented! . . . . .	26
<b>Mesh</b> . . . . .	27
<b>Model</b> . . . . .	29
<b>Models_Manager</b>	
Manager for all models in a scene . . . . .	32
<b>Particle_Container</b> . . . . .	34
<b>Physics_Manager</b> . . . . .	37
<b>PointLight</b>	
A point light . . . . .	38
<b>Scene_Container</b> . . . . .	39
<b>Scene_Manager</b> . . . . .	42

<b>Shader_Factory</b> . . . . .	43
<b>Shadow_Manager</b>	
Manager for shadows implemented as a Singleton . . . . .	44
<b>SpotLight</b>	
A spot light . . . . .	46
<b>Texture</b>	
A texture. This struct is currently not used! . . . . .	47
<b>Transform</b>	
<b>Transform</b> (p. 47) composed of Scale, Rotation (as a quaternion), and Translation. Transformation is applied in the order: Scale -> Rotate -> Translate . . . . .	47
<b>VboIndexer</b> . . . . .	51
<b>VertexFormat</b>	
The vertex buffer format that is sent directly to the shader . . . . .	52
<b>WindowInfo</b>	
Information about the game window . . . . .	53

## Chapter 4

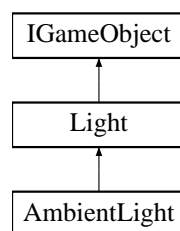
# Class Documentation

### 4.1 AmbientLight Class Reference

An ambient light. Uniformly lights up everything.

```
#include <Light.h>
```

Inheritance diagram for AmbientLight:



#### Public Member Functions

- **AmbientLight** (glm::vec3 **color**, float strength=1.0)
- virtual void **Draw** (GLuint) override final

*Sets up the light uniform location id's from the program. Each subclass must then send their corresponding params to the shader.*

#### Additional Inherited Members

##### 4.1.1 Detailed Description

An ambient light. Uniformly lights up everything.

##### 4.1.2 Constructor & Destructor Documentation

4.1.2.1 **AmbientLight::AmbientLight** ( glm::vec3 *color*, float *strength* = 1.0 )

##### 4.1.3 Member Function Documentation

4.1.3.1 void **AmbientLight::Draw** ( GLuint *program* ) [final], [override], [virtual]

Sets up the light uniform location id's from the program. Each subclass must then send their corresponding params to the shader.

## Parameters

<i>program</i>	The shader program.
----------------	---------------------

Reimplemented from **Light** (p. 22).

## 4.2 Camera Class Reference

The game camera implemented as a Singleton. Primary use is to generate Model-View-Projection matrices.

```
#include <Camera.h>
```

### Public Member Functions

- **Camera** (**Camera** const &)=delete
- void **operator=** (**Camera** const &)=delete

### Static Public Member Functions

- static **Camera** & **GetInstance** ()
- static glm::mat4 **GetProjectionMatrix** ()  
*Gets the camera projection matrix.*
- static glm::mat4 **GetViewMatrix** ()  
*Gets the camera view matrix.*
- static glm::vec3 **GetEyeDirection** ()  
*Gets the direction the camera is looking at.*
- static void **resizeWindow** (float width, float height)
- static void **ComputeMatrices** ()

### Static Public Attributes

- static float **fov** = 45.0f  
*The field of view.*
- static float **aspect**  
*The aspect ratio.*
- static float **zNear**
- static float **zFar**

#### 4.2.1 Detailed Description

The game camera implemented as a Singleton. Primary use is to generate Model-View-Projection matrices.



## 4.2.2 Constructor & Destructor Documentation

4.2.2.1 `Camera::Camera ( Camera const & ) [delete]`

## 4.2.3 Member Function Documentation

4.2.3.1 `void Camera::ComputeMatrices ( ) [static]`

4.2.3.2 `static glm::vec3 Camera::GetEyeDirection ( ) [static]`

Gets the direction the camera is looking at.

### Returns

The eye direction.

4.2.3.3 `static Camera& Camera::GetInstance ( ) [inline],[static]`

4.2.3.4 `static glm::mat4 Camera::GetProjectionMatrix ( ) [static]`

Gets the camera projection matrix.

### Returns

The projection matrix.

4.2.3.5 `static glm::mat4 Camera::GetViewMatrix ( ) [static]`

Gets the camera view matrix.

### Returns

The view matrix.

4.2.3.6 `void Camera::operator= ( Camera const & ) [delete]`

4.2.3.7 `void Camera::resizeWindow ( float width, float height ) [static]`

## 4.2.4 Member Data Documentation

4.2.4.1 `float Camera::aspect [static]`

The aspect ratio.

4.2.4.2 `float Camera::fov = 45.0f` `[static]`

The field of view.

4.2.4.3 `float Camera::zFar` `[static]`

4.2.4.4 `float Camera::zNear` `[static]`

## 4.3 ContextInfo Struct Reference

```
#include <ContextInfo.h>
```

### Public Member Functions

- **ContextInfo** ()
- **ContextInfo** (int **major\_version**, int **minor\_version**, bool **core**)
- void **operator=** (const **ContextInfo** &info)

### Public Attributes

- int **major\_version**
- int **minor\_version**
- bool **core**

### 4.3.1 Constructor & Destructor Documentation

4.3.1.1 `ContextInfo::ContextInfo ( )` `[inline]`

4.3.1.2 `ContextInfo::ContextInfo ( int major_version, int minor_version, bool core )` `[inline]`

### 4.3.2 Member Function Documentation

4.3.2.1 `void ContextInfo::operator= ( const ContextInfo & info )` `[inline]`

### 4.3.3 Member Data Documentation

4.3.3.1 `bool ContextInfo::core`

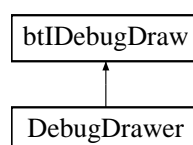
4.3.3.2 `int ContextInfo::major_version`

4.3.3.3 `int ContextInfo::minor_version`

## 4.4 DebugDrawer Class Reference

```
#include <DebugDrawer.h>
```

Inheritance diagram for DebugDrawer:



## Public Member Functions

- **DebugDrawer** ()
- **~DebugDrawer** ()
- void **drawLine** (const btVector3 &from, const btVector3 &to, const btVector3 &color) override
- void **drawContactPoint** (const btVector3 &PointOnB, const btVector3 &normalOnB, btScalar distance, int lifeTime, const btVector3 &color) override
- void **reportErrorWarning** (const char \*warningString) override
- void **draw3dText** (const btVector3 &location, const char \*textString) override
- void **setDebugMode** (int debugMode) override
- int **getDebugMode** () const override
- void **clearBuffers** ()
- void **render** ()

### 4.4.1 Constructor & Destructor Documentation

4.4.1.1 `DebugDrawer::DebugDrawer ( )`

4.4.1.2 `DebugDrawer::~~DebugDrawer ( )`

### 4.4.2 Member Function Documentation

4.4.2.1 `void DebugDrawer::clearBuffers ( )`

4.4.2.2 `void DebugDrawer::draw3dText ( const btVector3 & location, const char * textString )` [override]

4.4.2.3 `void DebugDrawer::drawContactPoint ( const btVector3 & PointOnB, const btVector3 & normalOnB, btScalar distance, int lifeTime, const btVector3 & color )` [override]

4.4.2.4 `void DebugDrawer::drawLine ( const btVector3 & from, const btVector3 & to, const btVector3 & color )` [override]

4.4.2.5 `int DebugDrawer::getDebugMode ( ) const` [override]

4.4.2.6 `void DebugDrawer::render ( )`

4.4.2.7 `void DebugDrawer::reportErrorWarning ( const char * warningString )` [override]

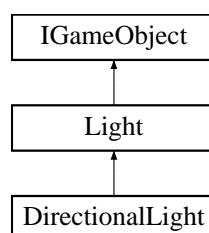
4.4.2.8 `void DebugDrawer::setDebugMode ( int debugMode )` [override]

## 4.5 DirectionalLight Class Reference

A directional light. Comes from "infinity" and lights up anything in the direction it is coming from.

```
#include <Light.h>
```

Inheritance diagram for DirectionalLight:



## Public Member Functions

- **DirectionalLight** (glm::vec3 **color**, glm::vec3 **position**, glm::vec3 **halfVector**)
- virtual void **Draw** (GLuint) override final  
*Sets up the light uniform location id's from the program. Each subclass must then send their corresponding params to the shader.*
- virtual void **DrawShadow** (GLuint) override final  
*Sends shadow information. Each subclass must implement this functionality!*

## Additional Inherited Members

### 4.5.1 Detailed Description

A directional light. Comes from "infinity" and lights up anything in the direction it is coming from.

### 4.5.2 Constructor & Destructor Documentation

4.5.2.1 **DirectionalLight::DirectionalLight** ( glm::vec3 *color*, glm::vec3 *position*, glm::vec3 *halfVector* )

### 4.5.3 Member Function Documentation

4.5.3.1 void **DirectionalLight::Draw** ( GLuint *program* ) [final],[override],[virtual]

Sets up the light uniform location id's from the program. Each subclass must then send their corresponding params to the shader.

#### Parameters

<i>program</i>	The shader program.
----------------	---------------------

Reimplemented from **Light** (p. 22).

4.5.3.2 void **DirectionalLight::DrawShadow** ( GLuint ) [final],[override],[virtual]

Sends shadow information. Each subclass must implement this functionality!

#### Parameters

<i>program</i>	The shader program.
----------------	---------------------

Reimplemented from **Light** (p. 23).

## 4.6 DVertexFormat Struct Reference

```
#include <DebugDrawer.h>
```

## Public Member Functions

- **DVertexFormat** (const glm::vec3 &inPos, const glm::vec3 &inColor)

## Public Attributes

- glm::vec3 **position**
- glm::vec3 **color**

### 4.6.1 Constructor & Destructor Documentation

4.6.1.1 **DVertexFormat::DVertexFormat** ( const glm::vec3 & *inPos*, const glm::vec3 & *inColor* ) `[inline]`

### 4.6.2 Member Data Documentation

4.6.2.1 glm::vec3 **DVertexFormat::color**

4.6.2.2 glm::vec3 **DVertexFormat::position**

## 4.7 FrameBufferInfo Struct Reference

```
#include <FrameBufferInfo.h>
```

## Public Member Functions

- **FrameBufferInfo** ()
- **FrameBufferInfo** (bool color, bool depth, bool stencil, bool **msaa**)

## Public Attributes

- unsigned int **flags**
- bool **msaa**

### 4.7.1 Constructor & Destructor Documentation

4.7.1.1 **FrameBufferInfo::FrameBufferInfo** ( ) `[inline]`

4.7.1.2 **FrameBufferInfo::FrameBufferInfo** ( bool *color*, bool *depth*, bool *stencil*, bool *msaa* ) `[inline]`

### 4.7.2 Member Data Documentation

4.7.2.1 unsigned int **FrameBufferInfo::flags**

4.7.2.2 bool **FrameBufferInfo::msaa**

## 4.8 GameObjectsBuilder Class Reference

```
#include <GameObjectsBuilder.h>
```

## Public Member Functions

- **GameObjectsBuilder** ()
- **~GameObjectsBuilder** ()
- **GameObjectsBuilder** & **addModel** (const std::string modelPath, const std::string texturePath=std::string())
- **GameObjectsBuilder** & **copyModel** ()
- **GameObjectsBuilder** & **setPosition** (glm::vec3)
- **GameObjectsBuilder** & **setScale** (float)
- **GameObjectsBuilder** & **setRotation** (float angleX, float angleY, float angleZ)
- **GameObjectsBuilder** & **setDiffuse** (const std::string &)
- **GameObjectsBuilder** & **setNormal** (const std::string &)
- **GameObjectsBuilder** & **setSpecular** (const std::string &)
- **GameObjectsBuilder** & **addRigidBody** (float mass)
- **GameObjectsBuilder** & **lockUpright** ()
- **GameObjectsBuilder** & **addParticleSystem** (const std::string &texturePath)
- **GameObjectsBuilder** & **setParticleCount** (int)
- **GameObjectsBuilder** & **setParticleSpawnRate** (int)
- **GameObjectsBuilder** & **setParticleLife** (float)
- **GameObjectsBuilder** & **addLight** (GameObjectType type)
- **GameObjectsBuilder** & **setColor** (glm::vec3 color)
- **GameObjectsBuilder** & **setHalfVector** (glm::vec3 halfVector)
- **GameObjectsBuilder** & **setCastsShadows** (bool)
- **GameObjectsBuilder** & **setStrength** (float strength)
- **GameObjectsBuilder** & **setAttenuation** (float constant, float linear, float quadratic)
- **GameObjectsBuilder** & **setConeDirection** (glm::vec3 coneDirection)
- **GameObjectsBuilder** & **setSpotCutoff** (float cutoff)
- **GameObjectsBuilder** & **setSpotExponent** (float exponent)
- **Models\_Manager** \* **getResult** () const

### 4.8.1 Constructor & Destructor Documentation

4.8.1.1 **GameObjectsBuilder::GameObjectsBuilder** ( )

4.8.1.2 **GameObjectsBuilder::~~GameObjectsBuilder** ( )

### 4.8.2 Member Function Documentation

4.8.2.1 **GameObjectsBuilder** & **GameObjectsBuilder::addLight** ( *GameObjectType type* )

4.8.2.2 **GameObjectsBuilder** & **GameObjectsBuilder::addModel** ( *const std::string modelPath*, *const std::string texturePath*  
= *std::string()* )

4.8.2.3 **GameObjectsBuilder** & **GameObjectsBuilder::addParticleSystem** ( *const std::string & texturePath* )

4.8.2.4 **GameObjectsBuilder** & **GameObjectsBuilder::addRigidBody** ( *float mass* )

4.8.2.5 **GameObjectsBuilder** & **GameObjectsBuilder::copyModel** ( )

4.8.2.6 **Models\_Manager** \* **GameObjectsBuilder::getResult** ( ) const

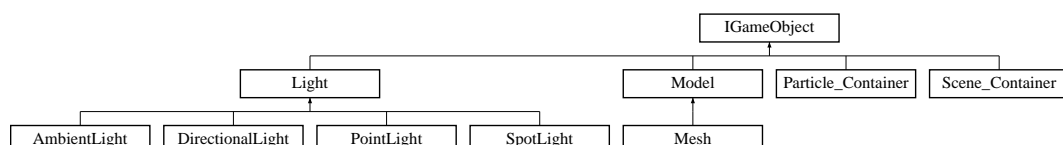
- 4.8.2.7 `GameObjectsBuilder & GameObjectsBuilder::lockUpright ( )`
- 4.8.2.8 `GameObjectsBuilder & GameObjectsBuilder::setAttenuation ( float constant, float linear, float quadratic )`
- 4.8.2.9 `GameObjectsBuilder & GameObjectsBuilder::setCastsShadows ( bool castsShadow )`
- 4.8.2.10 `GameObjectsBuilder & GameObjectsBuilder::setColor ( glm::vec3 color )`
- 4.8.2.11 `GameObjectsBuilder & GameObjectsBuilder::setConeDirection ( glm::vec3 coneDirection )`
- 4.8.2.12 `GameObjectsBuilder & GameObjectsBuilder::setDiffuse ( const std::string & path )`
- 4.8.2.13 `GameObjectsBuilder & GameObjectsBuilder::setHalfVector ( glm::vec3 halfVector )`
- 4.8.2.14 `GameObjectsBuilder & GameObjectsBuilder::setNormal ( const std::string & path )`
- 4.8.2.15 `GameObjectsBuilder & GameObjectsBuilder::setParticleCount ( int n )`
- 4.8.2.16 `GameObjectsBuilder & GameObjectsBuilder::setParticleLife ( float seconds )`
- 4.8.2.17 `GameObjectsBuilder & GameObjectsBuilder::setParticleSpawnRate ( int rate )`
- 4.8.2.18 `GameObjectsBuilder & GameObjectsBuilder::setPosition ( glm::vec3 pos )`
- 4.8.2.19 `GameObjectsBuilder & GameObjectsBuilder::setRotation ( float angleX, float angleY, float angleZ )`
- 4.8.2.20 `GameObjectsBuilder & GameObjectsBuilder::setScale ( float scale )`
- 4.8.2.21 `GameObjectsBuilder & GameObjectsBuilder::setSpecular ( const std::string & path )`
- 4.8.2.22 `GameObjectsBuilder & GameObjectsBuilder::setSpotCutoff ( float cutoff )`
- 4.8.2.23 `GameObjectsBuilder & GameObjectsBuilder::setSpotExponent ( float exponent )`
- 4.8.2.24 `GameObjectsBuilder & GameObjectsBuilder::setStrength ( float strength )`

## 4.9 IGameObject Class Reference

A Game Object. All objects that are to be rendered will inherit this class.

```
#include <IGameObject.h>
```

Inheritance diagram for IGameObject:



## Public Member Functions

- virtual `~IGameObject ()`  
*Destructor that deletes the unique shader program and shadow program belonging to the object.*
- virtual void **Draw** (GLuint shader)=0  
*Draws via the given shader.*
- virtual void **DrawShadow** (GLuint shader)=0  
*Draw shadows via the given shader.*
- virtual void **Update** ()=0  
*Updates this object.*
- virtual void **Destroy** ()=0  
*Destroys this object.*
- virtual void **SetProgram** (GLuint shaderName)  
*Sets the main shader program corresponding to this object.*
- virtual void **SetShadowProgram** (GLuint shaderName)  
*Sets the shadow pass shader program corresponding to this object.*
- virtual GLuint **GetProgram** ()  
*Gets the object's main shader program.*
- virtual GLuint **GetShadowProgram** ()  
*Gets the object's shadow program.*
- virtual void **SetTexture** (const std::string &textureName, const TextureType &textureType, const GLuint &texture)=0  
*Adds a texture to an object.*
- virtual const GLuint **GetTexture** (const std::string &textureName) const =0  
*Gets a texture.*

## Protected Attributes

- GLuint **program** = 0
- GLuint **shadowProgram** = 0

### 4.9.1 Detailed Description

A Game Object. All objects that are to be rendered will inherit this class.

### 4.9.2 Constructor & Destructor Documentation

#### 4.9.2.1 IGameObject::~~IGameObject ( ) [inline], [virtual]

Destructor that deletes the unique shader program and shadow program belonging to the object.

### 4.9.3 Member Function Documentation

#### 4.9.3.1 void IGameObject::Destroy ( ) [pure virtual]

Destroys this object.

Implemented in **Light** (p. 22), **Scene\_Container** (p. 40), **Model** (p. 30), and **Particle\_Container** (p. 35).

#### 4.9.3.2 void IGameObject::Draw ( GLuint *shader* ) [pure virtual]

Draws via the given shader.



## Parameters

<i>shader</i>	The shader.
---------------	-------------

Implemented in **SpotLight** (p. 47), **PointLight** (p. 38), **AmbientLight** (p. 7), **DirectionalLight** (p. 12), **Light** (p. 22), **Scene\_Container** (p. 40), **Mesh** (p. 28), and **Particle\_Container** (p. 35).

4.9.3.3 `void IGameObject::DrawShadow ( GLuint shader )` [pure virtual]

Draw shadows via the given shader.

Implemented in **DirectionalLight** (p. 12), **Light** (p. 23), **Scene\_Container** (p. 40), **Mesh** (p. 29), and **Particle\_Container** (p. 36).

4.9.3.4 `GLuint IGameObject::GetProgram ( )` [inline],[virtual]

Gets the object's main shader program.

## Returns

The main shader program.

4.9.3.5 `GLuint IGameObject::GetShadowProgram ( )` [inline],[virtual]

Gets the object's shadow program.

## Returns

The shadow program.

4.9.3.6 `const GLuint IGameObject::GetTexture ( const std::string & textureName ) const` [pure virtual]

Gets a texture.

## Parameters

<i>textureName</i>	The name of the texture.
--------------------	--------------------------

## Returns

The texture.

Implemented in **Light** (p. 23), **Scene\_Container** (p. 40), **Model** (p. 30), and **Particle\_Container** (p. 36).

**4.9.3.7** void `IGameObject::SetProgram` ( GLuint *shaderName* ) [inline],[virtual]

Sets the main shader program corresponding to this object.

Author

Harry

Date

4/14/2016

Parameters

<i>shaderName</i>	Name of the shader.
-------------------	---------------------

Reimplemented in **Scene\_Container** (p. 41), and **Model** (p. 31).

**4.9.3.8** void `IGameObject::SetShadowProgram` ( GLuint *shaderName* ) [inline],[virtual]

Sets the shadow pass shader program corresponding to this object.

Parameters

<i>shaderName</i>	Name of the shader.
-------------------	---------------------

Reimplemented in **Scene\_Container** (p. 41), and **Model** (p. 31).

**4.9.3.9** void `IGameObject::SetTexture` ( const std::string & *textureName*, const TextureType & *textureType*, const GLuint & *texture* ) [pure virtual]

Adds a texture to an object.

Parameters

<i>textureName</i>	Name of the texture.
<i>textureType</i>	Type of the texture.
<i>texture</i>	The texture id.

Implemented in **Light** (p. 24), **Scene\_Container** (p. 42), **Model** (p. 32), and **Particle\_Container** (p. 36).

**4.9.3.10** void `IGameObject::Update` ( ) [pure virtual]

Updates this object.

Implemented in **Light** (p. 24), **Scene\_Container** (p. 42), **Mesh** (p. 29), and **Particle\_Container** (p. 37).

#### 4.9.4 Member Data Documentation

4.9.4.1 GLuint IGameObject::program = 0 [protected]

4.9.4.2 GLuint IGameObject::shadowProgram = 0 [protected]

### 4.10 Init\_GLEW Class Reference

Helper class to initialize GLEW.

```
#include <Init_GLEW.h>
```

#### Public Member Functions

- **Init\_GLEW** ()
- **~Init\_GLEW** ()

#### Static Public Member Functions

- static void **Init** ()

#### 4.10.1 Detailed Description

Helper class to initialize GLEW.

#### 4.10.2 Constructor & Destructor Documentation

4.10.2.1 Init\_GLEW::Init\_GLEW ( )

4.10.2.2 Init\_GLEW::~~Init\_GLEW ( )

#### 4.10.3 Member Function Documentation

4.10.3.1 void Init\_GLEW::Init ( ) [static]

### 4.11 Init\_GLUT Class Reference

Helper class to initialize GLUT and setup listener callbacks.

```
#include <Init_GLUT.h>
```

#### Public Member Functions

- void **enterFullscreen** ()
- void **exitFullscreen** ()

## Static Public Member Functions

- static void **init** (const **WindowInfo** &, const **ContextInfo** &, const **FrameBufferInfo** &)  
*Initializes GLUT with a window.*
- static void **run** ()
- static void **close** ()
- static void **printOpenGLInfo** (const **WindowInfo** &window, const **ContextInfo** &context)
- static void **setListener** (**Scene\_Manager** \*&iListener)

### 4.11.1 Detailed Description

Helper class to initialize GLUT and setup listener callbacks.

### 4.11.2 Member Function Documentation

4.11.2.1 void Init\_GLUT::close ( ) [static]

4.11.2.2 void Init\_GLUT::enterFullscreen ( )

4.11.2.3 void Init\_GLUT::exitFullscreen ( )

4.11.2.4 static void Init\_GLUT::init ( const WindowInfo & windowInfo, const ContextInfo & contextInfo, const FrameBufferInfo & frameBufferInfo ) [static]

Initializes GLUT with a window.

4.11.2.5 void Init\_GLUT::printOpenGLInfo ( const WindowInfo & window, const ContextInfo & context ) [static]

4.11.2.6 void Init\_GLUT::run ( ) [static]

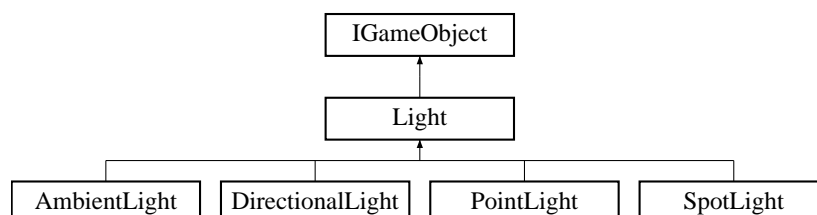
4.11.2.7 void Init\_GLUT::setListener ( Scene\_Manager \*&iListener ) [static]

## 4.12 Light Class Reference

A light is a special class of Game Object that mostly just sends uniform data to the shaders. Lights are fundamental to the shading pipeline! Without a light, nothing will show up in the scene.

```
#include <Light.h>
```

Inheritance diagram for Light:



## Classes

- struct **LightUniformLocations**

*Holds the various uniform location ID's of the **Light** (p. 20).*

## Public Member Functions

- **Light** ()
- **~Light** ()
- virtual void **Draw** (GLuint) override  
*Sets up the light uniform location id's from the program. Each subclass must then send their corresponding params to the shader.*
- virtual void **DrawShadow** (GLuint) override  
*Sends shadow information. Each subclass must implement this functionality!*
- virtual void **Update** () override  
*Updates the light. This currently does nothing! **Draw()** (p. 22) does update the light.*
- virtual void **Destroy** () override  
*Destroys the light. This currently only turns off the enabled flag!*
- virtual void **SetTexture** (const std::string &, const TextureType &, const GLuint &) override  
*Adds a texture to an object.*
- virtual const GLuint **GetTexture** (const std::string &) const override  
*Gets a texture.*
- bool **EnableShadows** ()  
*Enables shadow generation for this light.*
- bool **DisableShadows** ()  
*Disables shadow generation for this light.*
- void **SetAttenuation** (float constant=0.0f, float linear=1.0f, float quadratic=0.0f)  
*Sets the attenuation parameters. See this wiki article for more information about attenuation: [https://developer.valvesoftware.com/wiki/Constant-Linear-Quadratic\\_Falloff](https://developer.valvesoftware.com/wiki/Constant-Linear-Quadratic_Falloff).*

## Public Attributes

- glm::vec3 **ambient**  
*light's contribution to ambient light.*
- glm::vec3 **color**  
*The color/intensity of the light.*
- glm::vec3 **position**  
*location of light if isLocal is true. For directional light, it is the direction*
- glm::vec3 **halfVector**  
*?? direction of headlights for directional light ??*
- glm::vec3 **coneDirection**  
*spotlight cone direction.*
- float **spotCosCutoff**  
*how wide the spotlight is [0-1].*
- float **spotExponent**  
*control light fall-off in the spotlight.*
- float **constantAttenuation**  
*constant light fall-out*
- float **linearAttenuation**  
*linear light fall-off*
- float **quadraticAttenuation**  
*quadratic light fall-out*
- bool **isEnabled**  
*true if this object is enabled.*

## Protected Attributes

- GLuint **texture**
- unsigned int **\_id**
- bool **castsShadow**
- int **type**
- struct **Light::LightUniformLocations** **ids**

### 4.12.1 Detailed Description

A light is a special class of Game Object that mostly just sends uniform data to the shaders. Lights are fundamental to the shading pipeline! Without a light, nothing will show up in the scene.

### 4.12.2 Constructor & Destructor Documentation

#### 4.12.2.1 Light::Light ( )

#### 4.12.2.2 Light::~~Light ( ) [inline]

### 4.12.3 Member Function Documentation

#### 4.12.3.1 void Light::Destroy ( ) [override],[virtual]

Destroys the light. This currently only turns off the enabled flag!

Implements **IGameObject** (p. 16).

#### 4.12.3.2 bool Light::DisableShadows ( )

Disables shadow generation for this light.

#### Returns

true if it succeeds, false if it fails.

#### 4.12.3.3 void Light::Draw ( GLuint *program* ) [override],[virtual]

Sets up the light uniform location id's from the program. Each subclass must then send their corresponding params to the shader.

#### Parameters

<i>program</i>	The shader program.
----------------	---------------------

Implements **IGameObject** (p. 16).

Reimplemented in **SpotLight** (p. 47), **PointLight** (p. 38), **AmbientLight** (p. 7), and **DirectionalLight** (p. 12).

#### 4.12.3.4 void Light::DrawShadow ( GLuint ) [override],[virtual]

Sends shadow information. Each subclass must implement this functionality!

##### Parameters

<i>program</i>	The shader program.
----------------	---------------------

Implements **IGameObject** (p. 17).

Reimplemented in **DirectionalLight** (p. 12).

#### 4.12.3.5 bool Light::EnableShadows ( )

Enables shadow generation for this light.

##### Returns

true if it succeeds, false if it fails.

#### 4.12.3.6 const GLuint Light::GetTexture ( const std::string & textureName ) const [override],[virtual]

Gets a texture.

##### Parameters

<i>textureName</i>	The name of the texture.
--------------------	--------------------------

##### Returns

The texture.

Implements **IGameObject** (p. 17).

#### 4.12.3.7 void Light::SetAttenuation ( float constant = 0.0f, float linear = 1.0f, float quadratic = 0.0f )

Sets the attenuation parameters. See this wiki article for more information about attenuation: [https://developer.valvesoftware.com/wiki/Constant-Linear-Quadratic\\_Falloff](https://developer.valvesoftware.com/wiki/Constant-Linear-Quadratic_Falloff).

##### Author

Harry

##### Date

4/14/2016

## Parameters

<i>constant</i>	The constant attenuation factor.
<i>linear</i>	The linear attenuation factor.
<i>quadratic</i>	The quadratic attenuation factor.

4.12.3.8 `void Light::SetTexture ( const std::string & textureName, const TextureType & textureType, const GLuint & texture )`  
`[override],[virtual]`

Adds a texture to an object.

## Parameters

<i>textureName</i>	Name of the texture.
<i>textureType</i>	Type of the texture.
<i>texture</i>	The texture id.

Implements **IGameObject** (p. 18).

4.12.3.9 `void Light::Update ( )` `[override],[virtual]`

Updates the light. This currently does nothing! **Draw()** (p. 22) does update the light.

Implements **IGameObject** (p. 18).

## 4.12.4 Member Data Documentation

4.12.4.1 `unsigned int Light::_id` `[protected]`

4.12.4.2 `glm::vec3 Light::ambient`

light's contribution to ambient light.

4.12.4.3 `bool Light::castsShadow` `[protected]`

4.12.4.4 `glm::vec3 Light::color`

The color/intensity of the light.

4.12.4.5 `glm::vec3 Light::coneDirection`

spotlight cone direction.



4.12.4.6 float **Light::constantAttenuation**

constant light fall-out

4.12.4.7 glm::vec3 **Light::halfVector**

?? direction of headlights for directional light ??

4.12.4.8 struct **Light::LightUniformLocations** **Light::ids** [protected]4.12.4.9 bool **Light::isEnabled**

true if this object is enabled.

4.12.4.10 float **Light::linearAttenuation**

linear light fall-off

4.12.4.11 glm::vec3 **Light::position**

location of light if isLocal is true. For directional light, it is the direction

4.12.4.12 float **Light::quadraticAttenuation**

quadratic light fall-out

4.12.4.13 float **Light::spotCosCutoff**

how wide the spotlight is [0-1].

4.12.4.14 float **Light::spotExponent**

control light fall-off in the spotlight.

4.12.4.15 GLuint **Light::texture** [protected]4.12.4.16 int **Light::type** [protected]

## 4.13 **Light::LightUniformLocations** Struct Reference

Holds the various uniform location ID's of the **Light** (p. 20).

```
#include <Light.h>
```

## Public Attributes

- GLuint **isEnabled**
- GLuint **type**
- GLuint **ambient**
- GLuint **color**
- GLuint **position**
- GLuint **halfVector**
- GLuint **coneDirection**
- GLuint **spotCosCutoff**
- GLuint **spotExponent**
- GLuint **constantAttenuation**
- GLuint **linearAttenuation**
- GLuint **quadraticAttenuation**

### 4.13.1 Detailed Description

Holds the various uniform location ID's of the **Light** (p. 20).

### 4.13.2 Member Data Documentation

4.13.2.1 GLuint Light::LightUniformLocations::ambient

4.13.2.2 GLuint Light::LightUniformLocations::color

4.13.2.3 GLuint Light::LightUniformLocations::coneDirection

4.13.2.4 GLuint Light::LightUniformLocations::constantAttenuation

4.13.2.5 GLuint Light::LightUniformLocations::halfVector

4.13.2.6 GLuint Light::LightUniformLocations::isEnabled

4.13.2.7 GLuint Light::LightUniformLocations::linearAttenuation

4.13.2.8 GLuint Light::LightUniformLocations::position

4.13.2.9 GLuint Light::LightUniformLocations::quadraticAttenuation

4.13.2.10 GLuint Light::LightUniformLocations::spotCosCutoff

4.13.2.11 GLuint Light::LightUniformLocations::spotExponent

4.13.2.12 GLuint Light::LightUniformLocations::type

## 4.14 Material Struct Reference

A material. XXX Not yet implemented!

```
#include <Material.h>
```

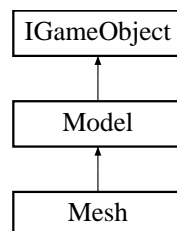
### 4.14.1 Detailed Description

A material. XXX Not yet implemented!

## 4.15 Mesh Class Reference

```
#include <Mesh.h>
```

Inheritance diagram for Mesh:



### Public Member Functions

- **Mesh** (const aiMesh \*, const aiMaterial \*, **Transform** \*)
- **Mesh** (const **Mesh** \*, **Transform** \*)
- **~Mesh** ()
- std::vector< **VertexFormat** > **GetVertices** () override final
- void **Create** ()
- void **DrawShadow** (GLuint) override final  
*Draw shadows via the given shader.*
- void **Draw** (GLuint) override final  
*Draws via the given shader.*
- void **Update** () override final  
*Updates this object.*
- std::vector< glm::vec3 > **GetPositionVertices** ()

### Public Attributes

- float **shininess**
- float **strength**

## Additional Inherited Members

### 4.15.1 Constructor & Destructor Documentation

4.15.1.1 `Mesh::Mesh ( const aiMesh * ai_mesh, const aiMaterial * ai_mat, Transform * transform )`

4.15.1.2 `Mesh::Mesh ( const Mesh * mesh, Transform * t )`

4.15.1.3 `Mesh::~~Mesh ( )`

### 4.15.2 Member Function Documentation

4.15.2.1 `void Mesh::Create ( )`

4.15.2.2 `void Mesh::Draw ( GLuint shader )` `[final]`, `[override]`, `[virtual]`

Draws via the given shader.

## Parameters

<i>shader</i>	The shader.
---------------	-------------

Implements **IGameObject** (p. 16).

4.15.2.3 `void Mesh::DrawShadow ( GLuint shader ) [final],[override],[virtual]`

Draw shadows via the given shader.

Implements **IGameObject** (p. 17).

4.15.2.4 `std::vector< glm::vec3 > Mesh::GetPositionVertices ( )`

4.15.2.5 `std::vector< VertexFormat > Mesh::GetVertices ( ) [final],[override],[virtual]`

Implements **Model** (p. 31).

4.15.2.6 `void Mesh::Update ( ) [final],[override],[virtual]`

Updates this object.

Implements **IGameObject** (p. 18).

### 4.15.3 Member Data Documentation

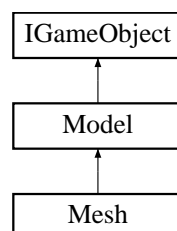
4.15.3.1 `float Mesh::shininess`

4.15.3.2 `float Mesh::strength`

## 4.16 Model Class Reference

```
#include <Model.h>
```

Inheritance diagram for Model:



## Public Member Functions

- **Model** ()
- **Model** (**Transform** \*t)
- virtual ~**Model** ()
- void **SetProgram** (GLuint) override  
*Sets the main shader program corresponding to this object.*
- void **SetShadowProgram** (GLuint) override  
*Sets the shadow pass shader program corresponding to this object.*
- void **Destroy** () override  
*Destroys this object.*
- virtual std::vector< **VertexFormat** > **GetVertices** ()=0
- GLuint **GetVao** () const
- const std::vector< GLuint > & **GetVbos** () const
- const GLuint **GetTexture** (const std::string &) const override  
*Gets a texture.*
- void **SetTexture** (const std::string &textureName, const TextureType &textureType, const GLuint &texture) override  
*Adds a texture to an object.*

## Protected Attributes

- **Transform** \* transform
- GLuint vao
- std::vector< GLuint > vbos
- std::vector< **Texture** > textures

## 4.16.1 Constructor & Destructor Documentation

4.16.1.1 **Model::Model** ( )

4.16.1.2 **Model::Model** ( **Transform** \* t )

4.16.1.3 **Model::~~Model** ( ) [virtual]

## 4.16.2 Member Function Documentation

4.16.2.1 void **Model::Destroy** ( ) [override],[virtual]

Destroys this object.

Implements **IGameObject** (p. 16).

4.16.2.2 const GLuint **Model::GetTexture** ( const std::string & textureName ) const [override],[virtual]

Gets a texture.

## Parameters

<i>textureName</i>	The name of the texture.
--------------------	--------------------------

## Returns

The texture.

Implements **IGameObject** (p. 17).

4.16.2.3 `GLuint Model::GetVao ( ) const`

4.16.2.4 `const std::vector< GLuint > & Model::GetVbos ( ) const`

4.16.2.5 `virtual std::vector<VertexFormat> Model::GetVertices ( ) [pure virtual]`

Implemented in **Mesh** (p. 29).

4.16.2.6 `void Model::SetProgram ( GLuint shaderName ) [override],[virtual]`

Sets the main shader program corresponding to this object.

## Author

Harry

## Date

4/14/2016

## Parameters

<i>shaderName</i>	Name of the shader.
-------------------	---------------------

Reimplemented from **IGameObject** (p. 18).

4.16.2.7 `void Model::SetShadowProgram ( GLuint shaderName ) [override],[virtual]`

Sets the shadow pass shader program corresponding to this object.

## Parameters

<i>shaderName</i>	Name of the shader.
-------------------	---------------------

Reimplemented from **IGameObject** (p. 18).

4.16.2.8 `void Model::SetTexture ( const std::string & textureName, const TextureType & textureType, const GLuint & texture )`  
`[override],[virtual]`

Adds a texture to an object.

#### Parameters

<i>textureName</i>	Name of the texture.
<i>textureType</i>	Type of the texture.
<i>texture</i>	The texture id.

Implements **IGameObject** (p. 18).

### 4.16.3 Member Data Documentation

4.16.3.1 `std::vector<Texture> Model::textures` `[protected]`

4.16.3.2 `Transform* Model::transform` `[protected]`

4.16.3.3 `GLuint Model::vao` `[protected]`

4.16.3.4 `std::vector<GLuint> Model::vbos` `[protected]`

## 4.17 Models\_Manager Class Reference

Manager for all models in a scene.

```
#include <Models_Manager.h>
```

### Public Member Functions

- **Models\_Manager** ()
- **~Models\_Manager** ()
- void **Draw** ()  
*The **Scene\_Manager** (p. 42) will call this method. Notifies all the **Scene\_Container** (p. 39), **Particle\_System**, and **Light** (p. 20) to draw.*
- void **DrawShadows** ()  
*The **Scene\_Manager** (p. 42) will call this method. Notifies all the **Scene\_Container** (p. 39), **Particle\_System**, and **Light** (p. 20) to draw shadows.*
- void **Update** ()  
*The **Scene\_Manager** (p. 42) will call this method. Notifies all the **Scene\_Container** (p. 39), **Particle\_System**, and **Light** (p. 20) to **Draw()** (p. 34).*
- void **addLight** (**Light** \*light)  
*Adds a light to the light list.*
- **Scene\_Container** \* **CreateModel** (const std::string &modelPath, const **Transform** &transform, const std::string &texturePath="", const TextureType type=Texture\_Diffuse)  
*Creates a model and adds it to the model list.*
- **Scene\_Container** \* **CreateModel** (const **Scene\_Container** \*&otherModel, const **Transform** &transform)  
*Copies a model and adds it to the model list.*
- **Particle\_Container** \* **CreateParticleSystem** (**Transform** transform, const std::string &texturePath)  
*Creates particle system and adds it to the particle list.*



### 4.17.1 Detailed Description

Manager for all models in a scene.

### 4.17.2 Constructor & Destructor Documentation

4.17.2.1 `Models_Manager::Models_Manager ( )`

4.17.2.2 `Models_Manager::~~Models_Manager ( )`

### 4.17.3 Member Function Documentation

4.17.3.1 `void Models_Manager::addLight ( Light * light )`

Adds a light to the light list.

#### Parameters

<i>light</i>	If non-null, the light.
--------------	-------------------------

4.17.3.2 `Scene_Container * Models_Manager::CreateModel ( const std::string & modelPath, const Transform & transform, const std::string & texturePath = " ", const TextureType type = Texture_Diffuse )`

Creates a model and adds it to the model list.

#### Parameters

<i>modelPath</i>	Full pathname to the object file.
<i>transform</i>	The transform.
<i>texturePath</i>	Full pathname to the texture file.
<i>type</i>	The texture type.

#### Returns

null if it fails, else the new model.

4.17.3.3 `Scene_Container * Models_Manager::CreateModel ( const Scene_Container *& otherModel, const Transform & transform )`

Copies a model and adds it to the model list.

#### Parameters

<i>otherModel</i>	The other model.
<i>transform</i>	The transform.

**Returns**

null if it fails, else the new model.

#### 4.17.3.4 **Particle\_Container \* Models\_Manager::CreateParticleSystem ( Transform *t*, const std::string & *texturePath* )**

Creates particle system and adds it to the particle list.

**Parameters**

<i>transform</i>	The <b>Transform</b> (p. 47)
<i>texturePath</i>	Full pathname to the texture file.

**Returns**

null if it fails, else the new particle system.

#### 4.17.3.5 **void Models\_Manager::Draw ( )**

The **Scene\_Manager** (p. 42) will call this method. Notifies all the **Scene\_Container** (p. 39), **Particle\_System**, and **Light** (p. 20) to draw.

#### 4.17.3.6 **void Models\_Manager::DrawShadows ( )**

The **Scene\_Manager** (p. 42) will call this method. Notifies all the **Scene\_Container** (p. 39), **Particle\_System**, and **Light** (p. 20) to draw shadows.

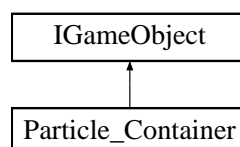
#### 4.17.3.7 **void Models\_Manager::Update ( )**

The **Scene\_Manager** (p. 42) will call this method. Notifies all the **Scene\_Container** (p. 39), **Particle\_System**, and **Light** (p. 20) to **Draw()** (p. 34).

## 4.18 Particle\_Container Class Reference

```
#include <Particle_Container.h>
```

Inheritance diagram for Particle\_Container:



## Public Member Functions

- **Particle\_Container** (**Transform** t, const std::string &texturePath, bool enable=true, int max\_particles=10000, int particle\_rate=1000, float particle\_max\_life=5)
- **~Particle\_Container** ()
- void **Draw** ()
- void **Draw** (GLuint) override  
*Draws via the given shader.*
- void **DrawShadow** ()
- void **DrawShadow** (GLuint) override  
*Draw shadows via the given shader.*
- void **Update** () override  
*Updates this object.*
- void **Destroy** () override  
*Destroys this object.*
- void **SetTexture** (const std::string &, const TextureType &, const GLuint &) override  
*Adds a texture to an object.*
- const GLuint **GetTexture** (const std::string &) const override  
*Gets a texture.*
- void **setMaxParticles** (int)
- void **setSpawnRate** (int)
- void **setMaxLife** (float)

## Public Attributes

- **Transform** transform

## Additional Inherited Members

### 4.18.1 Constructor & Destructor Documentation

4.18.1.1 **Particle\_Container::Particle\_Container** ( **Transform** t, const std::string & texturePath, bool enable = true, int max\_particles = 10000, int particle\_rate = 1000, float particle\_max\_life = 5 )

4.18.1.2 **Particle\_Container::~Particle\_Container** ( )

### 4.18.2 Member Function Documentation

4.18.2.1 **void Particle\_Container::Destroy** ( ) [override],[virtual]

Destroys this object.

Implements **IGameObject** (p. 16).

4.18.2.2 **void Particle\_Container::Draw** ( )

4.18.2.3 **void Particle\_Container::Draw** ( GLuint shader ) [override],[virtual]

Draws via the given shader.

## Parameters

<i>shader</i>	The shader.
---------------	-------------

Implements **IGameObject** (p. 16).

4.18.2.4 `void Particle_Container::DrawShadow ( ) [inline]`

4.18.2.5 `void Particle_Container::DrawShadow ( GLuint shader ) [inline],[override],[virtual]`

Draw shadows via the given shader.

Implements **IGameObject** (p. 17).

4.18.2.6 `const GLuint Particle_Container::GetTexture ( const std::string & textureName ) const [override],[virtual]`

Gets a texture.

## Parameters

<i>textureName</i>	The name of the texture.
--------------------	--------------------------

## Returns

The texture.

Implements **IGameObject** (p. 17).

4.18.2.7 `void Particle_Container::setMaxLife ( float life )`

4.18.2.8 `void Particle_Container::setMaxParticles ( int n )`

4.18.2.9 `void Particle_Container::setSpawnRate ( int rate )`

4.18.2.10 `void Particle_Container::SetTexture ( const std::string & textureName, const TextureType & textureType, const GLuint & texture ) [override],[virtual]`

Adds a texture to an object.

## Parameters

<i>textureName</i>	Name of the texture.
<i>textureType</i>	Type of the texture.
<i>texture</i>	The texture id.

Implements **IGameObject** (p. 18).

4.18.2.11 `void Particle_Container::Update ( ) [override],[virtual]`

Updates this object.

Implements **IGameObject** (p. 18).

### 4.18.3 Member Data Documentation

4.18.3.1 `Transform Particle_Container::transform`

## 4.19 Physics\_Manager Class Reference

```
#include <Physics_Manager.h>
```

### Public Member Functions

- `~Physics_Manager ( )`
- `void Step ( )`
- `void DrawDebug ( )`
- `void AddRigidBody (btRigidBody *)`
- `void AddConstraint (btTypedConstraint *)`

### Static Public Member Functions

- `static Physics_Manager * GetInstance ( )`

### 4.19.1 Constructor & Destructor Documentation

4.19.1.1 `Physics_Manager::~Physics_Manager ( )`

### 4.19.2 Member Function Documentation

4.19.2.1 `void Physics_Manager::AddConstraint ( btTypedConstraint * constraint )`

4.19.2.2 `void Physics_Manager::AddRigidBody ( btRigidBody * body )`

4.19.2.3 `void Physics_Manager::DrawDebug ( )`

4.19.2.4 `Physics_Manager * Physics_Manager::GetInstance ( ) [static]`

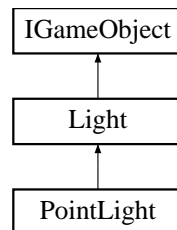
4.19.2.5 `void Physics_Manager::Step ( )`

## 4.20 PointLight Class Reference

A point light.

```
#include <Light.h>
```

Inheritance diagram for PointLight:



### Public Member Functions

- **PointLight** (glm::vec3 **color**, glm::vec3 **position**, float **constantAttenuation**=0.0f, float **linearAttenuation**=1.0f, float **quadraticAttenuation**=0.0f)
- virtual void **Draw** (GLuint) override final  
*Sets up the light uniform location id's from the program. Each subclass must then send their corresponding params to the shader.*

### Additional Inherited Members

#### 4.20.1 Detailed Description

A point light.

#### 4.20.2 Constructor & Destructor Documentation

**4.20.2.1** PointLight::PointLight ( glm::vec3 *color*, glm::vec3 *position*, float *constantAttenuation* = 0.0f, float *linearAttenuation* = 1.0f, float *quadraticAttenuation* = 0.0f )

#### 4.20.3 Member Function Documentation

**4.20.3.1** void PointLight::Draw ( GLuint *program* ) [final],[override],[virtual]

Sets up the light uniform location id's from the program. Each subclass must then send their corresponding params to the shader.

##### Parameters

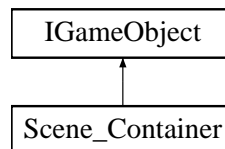
<i>program</i>	The shader program.
----------------	---------------------

Reimplemented from **Light** (p. 22).

## 4.21 Scene\_Container Class Reference

```
#include <Scene_Container.h>
```

Inheritance diagram for Scene\_Container:



### Public Member Functions

- **Scene\_Container** (const std::string &, **Transform**)
- **Scene\_Container** (const **Scene\_Container** \* &, **Transform**)
- **~Scene\_Container** ()
- void **Draw** ()
- void **Draw** (GLuint) override  
*Draws via the given shader.*
- void **DrawShadow** ()
- void **DrawShadow** (GLuint) override  
*Draw shadows via the given shader.*
- void **Update** () override  
*Updates this object.*
- void **Destroy** () override  
*Destroys this object.*
- void **SetProgram** (GLuint) override  
*Sets the main shader program corresponding to this object.*
- void **SetShadowProgram** (GLuint) override  
*Sets the shadow pass shader program corresponding to this object.*
- btRigidBody \* **getRigidBody** ()
- virtual void **InitRigidBody** (btScalar mass)
- const std::vector< **Mesh** \* > **GetMeshes** () const
- void **SetTexture** (const std::string &, const TextureType &, const GLuint &) override  
*Adds a texture to an object.*
- const GLuint **GetTexture** (const std::string &) const override  
*Gets a texture.*

### Public Attributes

- **Transform transform**

## Additional Inherited Members

### 4.21.1 Constructor & Destructor Documentation

4.21.1.1 `Scene_Container::Scene_Container ( const std::string & path, Transform transform )`

4.21.1.2 `Scene_Container::Scene_Container ( const Scene_Container * & other, Transform t )`

4.21.1.3 `Scene_Container::~Scene_Container ( )`

### 4.21.2 Member Function Documentation

4.21.2.1 `void Scene_Container::Destroy ( ) [override], [virtual]`

Destroys this object.

Implements **IGameObject** (p. 16).

4.21.2.2 `void Scene_Container::Draw ( )`

4.21.2.3 `void Scene_Container::Draw ( GLuint shader ) [override], [virtual]`

Draws via the given shader.

#### Parameters

<i>shader</i>	The shader.
---------------	-------------

Implements **IGameObject** (p. 16).

4.21.2.4 `void Scene_Container::DrawShadow ( )`

4.21.2.5 `void Scene_Container::DrawShadow ( GLuint shader ) [override], [virtual]`

Draw shadows via the given shader.

Implements **IGameObject** (p. 17).

4.21.2.6 `const std::vector< Mesh * > Scene_Container::GetMeshes ( ) const`

4.21.2.7 `btRigidBody * Scene_Container::getRigidBody ( )`

4.21.2.8 `const GLuint Scene_Container::GetTexture ( const std::string & textureName ) const [override], [virtual]`

Gets a texture.



## Parameters

<i>textureName</i>	The name of the texture.
--------------------	--------------------------

## Returns

The texture.

Implements **IGameObject** (p. 17).

4.21.2.9 void Scene\_Container::InitRigidBody ( *btScalar mass* ) [virtual]

**Scene\_Container** (p. 39) owns this method because I, Dylan, have made an executive decision to make lights bound to the laws of physics impossible; I see no point for it (except for flashlights maybe). Our lights will be static, so the initialization of rigid bodies can be the sole responsibility of the **Scene\_Container** (p. 39), because it knows about all of its own vertices.

TODO: Support more than just Convex Hull Shape for collision detection; Triangle **Mesh** (p. 27) Shape might be more appropriate for static game objects.

4.21.2.10 void Scene\_Container::SetProgram ( *GLuint shaderName* ) [override],[virtual]

Sets the main shader program corresponding to this object.

## Author

Harry

## Date

4/14/2016

## Parameters

<i>shaderName</i>	Name of the shader.
-------------------	---------------------

Reimplemented from **IGameObject** (p. 18).

4.21.2.11 void Scene\_Container::SetShadowProgram ( *GLuint shaderName* ) [override],[virtual]

Sets the shadow pass shader program corresponding to this object.

## Parameters

<i>shaderName</i>	Name of the shader.
-------------------	---------------------

Reimplemented from **IGameObject** (p. 18).

**4.21.2.12** `void Scene_Container::SetTexture ( const std::string & textureName, const TextureType & textureType, const GLuint & texture )` `[override],[virtual]`

Adds a texture to an object.

#### Parameters

<i>textureName</i>	Name of the texture.
<i>textureType</i>	Type of the texture.
<i>texture</i>	The texture id.

Implements **IGameObject** (p. 18).

**4.21.2.13** `void Scene_Container::Update ( )` `[override],[virtual]`

Updates this object.

Implements **IGameObject** (p. 18).

## 4.21.3 Member Data Documentation

**4.21.3.1** Transform Scene\_Container::transform

## 4.22 Scene\_Manager Class Reference

```
#include <Scene_Manager.h>
```

### Public Member Functions

- **Scene\_Manager** (std::string)
- **~Scene\_Manager** ()
- void **SetupScene** (const **GameObjectsBuilder** &)
- void **UpdatePass** () const
- void **ShadowPass** () const
- void **RenderPass** () const
- void **notifyBeginFrame** ()
- void **notifyDisplayFrame** ()
- void **notifyEndFrame** ()
- void **notifyReshape** (int width, int height, int p\_width, int p\_height)

### Static Public Member Functions

- static int **GetDeltaTime** ()
- static float **GetFPS** ()

### 4.22.1 Constructor & Destructor Documentation

4.22.1.1 `Scene_Manager::Scene_Manager ( std::string scene_name )`

4.22.1.2 `Scene_Manager::~~Scene_Manager ( )`

### 4.22.2 Member Function Documentation

4.22.2.1 `int Scene_Manager::GetDeltaTime ( ) [static]`

4.22.2.2 `float Scene_Manager::GetFPS ( ) [static]`

4.22.2.3 `void Scene_Manager::notifyBeginFrame ( )`

4.22.2.4 `void Scene_Manager::notifyDisplayFrame ( )`

4.22.2.5 `void Scene_Manager::notifyEndFrame ( )`

4.22.2.6 `void Scene_Manager::notifyReshape ( int width, int height, int p_width, int p_height )`

4.22.2.7 `void Scene_Manager::RenderPass ( ) const`

4.22.2.8 `void Scene_Manager::SetupScene ( const GameObjectsBuilder & gob )`

4.22.2.9 `void Scene_Manager::ShadowPass ( ) const`

4.22.2.10 `void Scene_Manager::UpdatePass ( ) const`

## 4.23 Shader\_Factory Class Reference

```
#include <Shader_Factory.h>
```

### Public Member Functions

- `~Shader_Factory` (void)
- `const GLuint & CreateProgram` (const std::string &shaderName, const std::string &VertexShaderFilename, const std::string &FragmentShaderFilename)
- `const void SetTextureShader` (IGameObject &model)
- `const GLuint & CreateDebugProgram` ()

### Static Public Member Functions

- `static Shader_Factory * GetInstance` ()

### 4.23.1 Constructor & Destructor Documentation

4.23.1.1 Shader\_Factory::~~Shader\_Factory ( void )

### 4.23.2 Member Function Documentation

4.23.2.1 const GLuint & Shader\_Factory::CreateDebugProgram ( )

4.23.2.2 const GLuint & Shader\_Factory::CreateProgram ( const std::string & *shaderName*, const std::string & *VertexShaderFilename*, const std::string & *FragmentShaderFilename* )

4.23.2.3 Shader\_Factory \* Shader\_Factory::GetInstance ( ) [static]

4.23.2.4 const void Shader\_Factory::SetTextureShader ( IGameObject & *model* )

## 4.24 Shadow\_Manager Class Reference

Manager for shadows implemented as a Singleton.

```
#include <Shadow_Manager.h>
```

### Public Member Functions

- GLuint **GetShadowMap** () const  
*Gets shadow map texture.*
- glm::mat4 **GetDepthMatrix** () const  
*Gets the 4x4 depth matrix.*
- void **SetDepthMatrix** (glm::mat4 DepthMatrix)  
*Sets the depth matrix.*
- void **BindForWriting** () const  
*Bind the depth texture for writing.*
- void **Unbind** () const  
*Unbinds the depth texture.*

### Static Public Member Functions

- static Shadow\_Manager \* **GetInstance** ()  
*Gets the instance.*

### Static Public Attributes

- static const int **DEPTH\_TEXTURE\_SIZE** = 1024  
*Size of the depth texture in pixels.*

### 4.24.1 Detailed Description

Manager for shadows implemented as a Singleton.

#### Author

Harry Gogonis

### 4.24.2 Member Function Documentation

#### 4.24.2.1 void Shadow\_Manager::BindForWriting ( ) const

Bind the depth texture for writing.

#### 4.24.2.2 glm::mat4 Shadow\_Manager::GetDepthMatrix ( ) const

Gets the 4x4 depth matrix.

#### Returns

The depth matrix.

#### 4.24.2.3 static Shadow\_Manager \* Shadow\_Manager::GetInstance ( ) [static]

Gets the instance.

#### Author

Harry

#### Returns

The Singleton instance

#### 4.24.2.4 GLuint Shadow\_Manager::GetShadowMap ( ) const

Gets shadow map texture.

#### Returns

The shadow map.

#### 4.24.2.5 void Shadow\_Manager::SetDepthMatrix ( glm::mat4 DepthMatrix )

Sets the depth matrix.

## Parameters

<i>DepthMatrix</i>	The depth matrix.
--------------------	-------------------

## 4.24.2.6 void Shadow\_Manager::Unbind ( ) const

Unbinds the depth texture.

## 4.24.3 Member Data Documentation

## 4.24.3.1 const int Shadow\_Manager::DEPTH\_TEXTURE\_SIZE = 1024 [static]

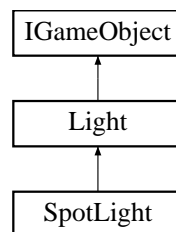
Size of the depth texture in pixels.

## 4.25 SpotLight Class Reference

A spot light.

```
#include <Light.h>
```

Inheritance diagram for SpotLight:



## Public Member Functions

- **SpotLight** (glm::vec3 **color**, glm::vec3 **position**, glm::vec3 **coneDirection**, float **spotCosCutoff**=0.4999f, float **spotExponent**=0.0f, float **constantAttenuation**=0.0f, float **linearAttenuation**=1.0f, float **quadraticAttenuation**=0.0f)
- virtual void **Draw** (GLuint) override final

*Sets up the light uniform location id's from the program. Each subclass must then send their corresponding params to the shader.*

## Additional Inherited Members

## 4.25.1 Detailed Description

A spot light.

## 4.25.2 Constructor & Destructor Documentation

4.25.2.1 `SpotLight::SpotLight ( glm::vec3 color, glm::vec3 position, glm::vec3 coneDirection, float spotCosCutoff = 0.99f, float spotExponent = 0.0f, float constantAttenuation = 0.0f, float linearAttenuation = 1.0f, float quadraticAttenuation = 0.0f )`

## 4.25.3 Member Function Documentation

4.25.3.1 `void SpotLight::Draw ( GLuint program )` [final],[override],[virtual]

Sets up the light uniform location id's from the program. Each subclass must then send their corresponding params to the shader.

### Parameters

<i>program</i>	The shader program.
----------------	---------------------

Reimplemented from **Light** (p.22).

## 4.26 Texture Struct Reference

A texture. This struct is currently not used!

```
#include <Texture.h>
```

### Public Attributes

- GLuint **id**
- std::string **name**
- TextureType **type**

### 4.26.1 Detailed Description

A texture. This struct is currently not used!

### 4.26.2 Member Data Documentation

4.26.2.1 GLuint Texture::id

4.26.2.2 std::string Texture::name

4.26.2.3 TextureType Texture::type

## 4.27 Transform Class Reference

**Transform** (p. 47) composed of Scale, Rotation (as a quaternion), and Translation. Transformation is applied in the order: Scale -> Rotate -> Translate.

```
#include <Transform.h>
```

## Public Member Functions

- **Transform** ()
- **Transform** (glm::vec3 **position**, float **scale**, glm::quat **rotation**)
- void **SetScale** (float **scale**)  
*Sets a scale.*
- void **SetPosition** (const glm::vec3 &**position**)  
*Sets a position.*
- void **SetRotation** (float angle, glm::vec3 axis)  
*Sets a rotation.*
- void **IncRotation** (float angle, glm::vec3 axis)  
*Increment rotation.*
- void **RotateX** (float angle)  
*Rotate along x axis.*
- void **RotateY** (float angle)  
*Rotate along y axis.*
- void **RotateZ** (float angle)  
*Rotate along z axis.*
- glm::mat4 **getTranslationMatrix** () const  
*Gets translation matrix.*
- glm::mat4 **getScaleMatrix** () const  
*Gets scale matrix.*
- glm::mat4 **getRotationMatrix** () const  
*Gets rotation matrix.*
- glm::mat4 **getTransformMatrix** () const  
*Gets transformation matrix. Transformation applied in the order: Scale -> Rotate -> Translate.*
- **Transform & operator=** (const btTransform &trans)  
*Assignment operator.*
- **~Transform** ()

## Public Attributes

- glm::vec3 **position**
- glm::quat **rotation**
- float **scale**

### 4.27.1 Detailed Description

**Transform** (p. 47) composed of Scale, Rotation (as a quaternion), and Translation. Transformation is applied in the order: Scale -> Rotate -> Translate.



## 4.27.2 Constructor & Destructor Documentation

### 4.27.2.1 Transform::Transform ( )

### 4.27.2.2 Transform::Transform ( glm::vec3 *position*, float *scale*, glm::quat *rotation* )

### 4.27.2.3 Transform::~~Transform ( )

## 4.27.3 Member Function Documentation

### 4.27.3.1 glm::mat4 Transform::getRotationMatrix ( ) const

Gets rotation matrix.

#### Returns

The rotation matrix.

### 4.27.3.2 glm::mat4 Transform::getScaleMatrix ( ) const

Gets scale matrix.

#### Returns

The scale matrix.

### 4.27.3.3 glm::mat4 Transform::getTransformMatrix ( ) const

Gets transformation matrix. Transformation applied in the order: Scale -> Rotate -> Translate.

#### Returns

The transform matrix.

### 4.27.3.4 glm::mat4 Transform::getTranslationMatrix ( ) const

Gets translation matrix.

#### Returns

The translation matrix.

### 4.27.3.5 void Transform::IncRotation ( float *angle*, glm::vec3 *axis* )

Increment rotation.

## Parameters

<i>angle</i>	The angle in degrees.
<i>axis</i>	The axis.

**4.27.3.6 Transform & Transform::operator= ( const btTransform & *trans* )**

Assignment operator.

## Parameters

<i>trans</i>	The transform from ASSIMP
--------------	---------------------------

## Returns

A shallow copy of this object.

**4.27.3.7 void Transform::RotateX ( float *angle* )**

Rotate along x axis.

## Parameters

<i>angle</i>	The angle in degrees.
--------------	-----------------------

**4.27.3.8 void Transform::RotateY ( float *angle* )**

Rotate along y axis.

## Parameters

<i>angle</i>	The angle in degrees.
--------------	-----------------------

**4.27.3.9 void Transform::RotateZ ( float *angle* )**

Rotate along z axis.

## Parameters

<i>angle</i>	The angle in degrees.
--------------	-----------------------

4.27.3.10 void Transform::SetPosition ( const glm::vec3 & *position* )

Sets a position.

Parameters

<i>position</i>	The position.
-----------------	---------------

4.27.3.11 void Transform::SetRotation ( float *angle*, glm::vec3 *axis* )

Sets a rotation.

Parameters

<i>angle</i>	The angle in degrees.
<i>axis</i>	The axis.

4.27.3.12 void Transform::SetScale ( float *scale* )

Sets a scale.

Parameters

<i>scale</i>	The scale.
--------------	------------

## 4.27.4 Member Data Documentation

4.27.4.1 glm::vec3 Transform::position

4.27.4.2 glm::quat Transform::rotation

4.27.4.3 float Transform::scale

## 4.28 VboIndexer Class Reference

```
#include <VboIndexer.h>
```

### Static Public Member Functions

- static void **indexVBO** (std::vector< glm::vec3 > &in\_vertices, std::vector< glm::vec3 > &in\_colors, std::vector< unsigned short > &out\_indices, std::vector< **DVertexFormat** > &out\_formats)

### 4.28.1 Member Function Documentation

- 4.28.1.1 `void VboIndexer::indexVBO ( std::vector< glm::vec3 > & in_vertices, std::vector< glm::vec3 > & in_colors, std::vector< unsigned short > & out_indices, std::vector< DVertexFormat > & out_formats ) [static]`

## 4.29 VertexFormat Struct Reference

The vertex buffer format that is sent directly to the shader.

```
#include <VertexFormat.h>
```

### Public Member Functions

- **VertexFormat** (const glm::vec3 &inPos, const glm::vec2 &inUV, const glm::vec3 &inNormal, const glm::vec3 &inTangent, const glm::vec3 &inBitangent)
- bool **operator<** (const **VertexFormat** that) const

### Public Attributes

- glm::vec3 **position**
- glm::vec2 **uv**
- glm::vec3 **normal**
- glm::vec3 **tangent**
- glm::vec3 **bitangent**

### Friends

- std::ostream & **operator<<** (std::ostream &os, const **VertexFormat** &v)

### 4.29.1 Detailed Description

The vertex buffer format that is sent directly to the shader.

### 4.29.2 Constructor & Destructor Documentation

- 4.29.2.1 `VertexFormat::VertexFormat ( const glm::vec3 & inPos, const glm::vec2 & inUV, const glm::vec3 & inNormal, const glm::vec3 & inTangent, const glm::vec3 & inBitangent ) [inline]`

### 4.29.3 Member Function Documentation

- 4.29.3.1 `bool VertexFormat::operator< ( const VertexFormat that ) const [inline]`

### 4.29.4 Friends And Related Function Documentation

4.29.4.1 `std::ostream& operator<< ( std::ostream & os, const VertexFormat & v )` [*friend*]

## 4.29.5 Member Data Documentation

4.29.5.1 `glm::vec3 VertexFormat::bitangent`

4.29.5.2 `glm::vec3 VertexFormat::normal`

4.29.5.3 `glm::vec3 VertexFormat::position`

4.29.5.4 `glm::vec3 VertexFormat::tangent`

4.29.5.5 `glm::vec2 VertexFormat::uv`

## 4.30 WindowInfo Struct Reference

Information about the game window.

```
#include <WindowInfo.h>
```

### Public Member Functions

- **WindowInfo** ()
- **WindowInfo** (std::string **name**, int **position\_x**, int **position\_y**, int **width**, int **height**, bool **isReshapable**)
- **WindowInfo** (const **WindowInfo** &windowInfo)
- void **operator=** (const **WindowInfo** &windowInfo)

### Public Attributes

- std::string **name**
- int **width**
- int **height**
- int **position\_x**
- int **position\_y**
- bool **isReshapable**

### 4.30.1 Detailed Description

Information about the game window.

Author

Harry

Date

4/14/2016

### 4.30.2 Constructor & Destructor Documentation

4.30.2.1 `WindowInfo::WindowInfo ( )` `[inline]`

4.30.2.2 `WindowInfo::WindowInfo ( std::string name, int position_x, int position_y, int width, int height, bool isReshapable )`  
`[inline]`

4.30.2.3 `WindowInfo::WindowInfo ( const WindowInfo & windowInfo )` `[inline]`

### 4.30.3 Member Function Documentation

4.30.3.1 `void WindowInfo::operator= ( const WindowInfo & windowInfo )` `[inline]`

### 4.30.4 Member Data Documentation

4.30.4.1 `int WindowInfo::height`

4.30.4.2 `bool WindowInfo::isReshapable`

4.30.4.3 `std::string WindowInfo::name`

4.30.4.4 `int WindowInfo::position_x`

4.30.4.5 `int WindowInfo::position_y`

4.30.4.6 `int WindowInfo::width`

# Index

- \_id
    - Light, 24
  - ~DebugDrawer
    - DebugDrawer, 11
  - ~GameObjectsBuilder
    - GameObjectsBuilder, 14
  - ~IGameObject
    - IGameObject, 16
  - ~Init\_GLEW
    - Init\_GLEW, 19
  - ~Light
    - Light, 22
  - ~Mesh
    - Mesh, 28
  - ~Model
    - Model, 30
  - ~Models\_Manager
    - Models\_Manager, 33
  - ~Particle\_Container
    - Particle\_Container, 35
  - ~Physics\_Manager
    - Physics\_Manager, 37
  - ~Scene\_Container
    - Scene\_Container, 40
  - ~Scene\_Manager
    - Scene\_Manager, 43
  - ~Shader\_Factory
    - Shader\_Factory, 44
  - ~Transform
    - Transform, 49
- AddConstraint
  - Physics\_Manager, 37
- addLight
  - GameObjectsBuilder, 14
  - Models\_Manager, 33
- addModel
  - GameObjectsBuilder, 14
- addParticleSystem
  - GameObjectsBuilder, 14
- AddRigidBody
  - Physics\_Manager, 37
- addRigidBody
  - GameObjectsBuilder, 14
- ambient
  - Light, 24
  - Light::LightUniformLocations, 26
- AmbientLight, 7
  - AmbientLight, 7
  - Draw, 7

- aspect
  - Camera, 9
- BindForWriting
  - Shadow\_Manager, 45
- bitangent
  - VertexFormat, 53
- Camera, 8
  - aspect, 9
  - Camera, 9
  - ComputeMatrices, 9
  - fov, 9
  - GetEyeDirection, 9
  - GetInstance, 9
  - GetProjectionMatrix, 9
  - GetViewMatrix, 9
  - operator=, 9
  - resizeWindow, 9
  - zFar, 10
  - zNear, 10
- castsShadow
  - Light, 24
- clearBuffers
  - DebugDrawer, 11
- close
  - Init\_GLUT, 20
- color
  - DVertexFormat, 13
  - Light, 24
  - Light::LightUniformLocations, 26
- ComputeMatrices
  - Camera, 9
- coneDirection
  - Light, 24
  - Light::LightUniformLocations, 26
- constantAttenuation
  - Light, 24
  - Light::LightUniformLocations, 26
- ContextInfo, 10
  - ContextInfo, 10
  - core, 10
  - major\_version, 10
  - minor\_version, 10
  - operator=, 10
- copyModel
  - GameObjectsBuilder, 14
- core
  - ContextInfo, 10
- Create

- Mesh, 28
- CreateDebugProgram
  - Shader\_Factory, 44
- CreateModel
  - Models\_Manager, 33
- CreateParticleSystem
  - Models\_Manager, 34
- CreateProgram
  - Shader\_Factory, 44
- DEPTH\_TEXTURE\_SIZE
  - Shadow\_Manager, 46
- DVertexFormat, 12
  - color, 13
  - DVertexFormat, 13
  - position, 13
- DebugDrawer, 10
  - ~DebugDrawer, 11
  - clearBuffers, 11
  - DebugDrawer, 11
  - draw3dText, 11
  - drawContactPoint, 11
  - drawLine, 11
  - getDebugMode, 11
  - render, 11
  - reportErrorWarning, 11
  - setDebugMode, 11
- Destroy
  - IGameObject, 16
  - Light, 22
  - Model, 30
  - Particle\_Container, 35
  - Scene\_Container, 40
- DirectionalLight, 11
  - DirectionalLight, 12
  - Draw, 12
  - DrawShadow, 12
- DisableShadows
  - Light, 22
- Draw
  - AmbientLight, 7
  - DirectionalLight, 12
  - IGameObject, 16
  - Light, 22
  - Mesh, 28
  - Models\_Manager, 34
  - Particle\_Container, 35
  - PointLight, 38
  - Scene\_Container, 40
  - SpotLight, 47
- draw3dText
  - DebugDrawer, 11
- drawContactPoint
  - DebugDrawer, 11
- DrawDebug
  - Physics\_Manager, 37
- drawLine
  - DebugDrawer, 11
- DrawShadow
  - DirectionalLight, 12
  - IGameObject, 17
  - Light, 23
  - Mesh, 29
  - Particle\_Container, 36
  - Scene\_Container, 40
- DrawShadows
  - Models\_Manager, 34
- EnableShadows
  - Light, 23
- enterFullscreen
  - Init\_GLUT, 20
- exitFullscreen
  - Init\_GLUT, 20
- flags
  - FrameBufferInfo, 13
- fov
  - Camera, 9
- FrameBufferInfo, 13
  - flags, 13
  - FrameBufferInfo, 13
  - msaa, 13
- GameObjectsBuilder, 13
  - ~GameObjectsBuilder, 14
  - addLight, 14
  - addModel, 14
  - addParticleSystem, 14
  - addRigidBody, 14
  - copyModel, 14
  - GameObjectsBuilder, 14
  - getResult, 14
  - lockUpright, 14
  - setAttenuation, 15
  - setCastsShadows, 15
  - setColor, 15
  - setConeDirection, 15
  - setDiffuse, 15
  - setHalfVector, 15
  - setNormal, 15
  - setParticleCount, 15
  - setParticleLife, 15
  - setParticleSpawnRate, 15
  - setPosition, 15
  - setRotation, 15
  - setScale, 15
  - setSpecular, 15
  - setSpotCutoff, 15
  - setSpotExponent, 15
  - setStrength, 15
- getDebugMode
  - DebugDrawer, 11
- GetDeltaTime
  - Scene\_Manager, 43
- GetDepthMatrix
  - Shadow\_Manager, 45
- GetEyeDirection



- Camera, 9
- GetFPS
  - Scene\_Manager, 43
- GetInstance
  - Camera, 9
  - Physics\_Manager, 37
  - Shader\_Factory, 44
  - Shadow\_Manager, 45
- GetMeshes
  - Scene\_Container, 40
- GetPositionVertices
  - Mesh, 29
- GetProgram
  - IGameObject, 17
- GetProjectionMatrix
  - Camera, 9
- getResult
  - GameObjectsBuilder, 14
- getRigidBody
  - Scene\_Container, 40
- getRotationMatrix
  - Transform, 49
- getScaleMatrix
  - Transform, 49
- GetShadowMap
  - Shadow\_Manager, 45
- GetShadowProgram
  - IGameObject, 17
- GetTexture
  - IGameObject, 17
  - Light, 23
  - Model, 30
  - Particle\_Container, 36
  - Scene\_Container, 40
- getTransformMatrix
  - Transform, 49
- getTranslationMatrix
  - Transform, 49
- GetVao
  - Model, 31
- GetVbos
  - Model, 31
- GetVertices
  - Mesh, 29
  - Model, 31
- GetViewMatrix
  - Camera, 9
- halfVector
  - Light, 25
  - Light::LightUniformLocations, 26
- height
  - WindowInfo, 54
- IGameObject, 15
  - ~IGameObject, 16
  - Destroy, 16
  - Draw, 16
  - DrawShadow, 17
  - GetProgram, 17
  - GetShadowProgram, 17
  - GetTexture, 17
  - program, 19
  - SetProgram, 17
  - SetShadowProgram, 18
  - SetTexture, 18
  - shadowProgram, 19
  - Update, 18
- id
  - Texture, 47
- ids
  - Light, 25
- IncRotation
  - Transform, 49
- indexVBO
  - VboIndexer, 52
- Init
  - Init\_GLEW, 19
- init
  - Init\_GLUT, 20
- Init\_GLEW, 19
  - ~Init\_GLEW, 19
  - Init, 19
  - Init\_GLEW, 19
- Init\_GLUT, 19
  - close, 20
  - enterFullscreen, 20
  - exitFullscreen, 20
  - init, 20
  - printOpenGLInfo, 20
  - run, 20
  - setListener, 20
- InitRigidBody
  - Scene\_Container, 41
- isEnabled
  - Light, 25
  - Light::LightUniformLocations, 26
- isReshapable
  - WindowInfo, 54
- Light, 20
  - \_id, 24
  - ~Light, 22
  - ambient, 24
  - castsShadow, 24
  - color, 24
  - coneDirection, 24
  - constantAttenuation, 24
  - Destroy, 22
  - DisableShadows, 22
  - Draw, 22
  - DrawShadow, 23
  - EnableShadows, 23
  - GetTexture, 23
  - halfVector, 25
  - ids, 25
  - isEnabled, 25
  - Light, 22

- linearAttenuation, 25
- position, 25
- quadraticAttenuation, 25
- SetAttenuation, 23
- SetTexture, 24
- spotCosCutoff, 25
- spotExponent, 25
- texture, 25
- type, 25
- Update, 24
- Light::LightUniformLocations, 25
  - ambient, 26
  - color, 26
  - coneDirection, 26
  - constantAttenuation, 26
  - halfVector, 26
  - isEnabled, 26
  - linearAttenuation, 26
  - position, 26
  - quadraticAttenuation, 26
  - spotCosCutoff, 26
  - spotExponent, 26
  - type, 26
- linearAttenuation
  - Light, 25
  - Light::LightUniformLocations, 26
- lockUpright
  - GameObjectsBuilder, 14
- major\_version
  - ContextInfo, 10
- Material, 26
- Mesh, 27
  - ~Mesh, 28
  - Create, 28
  - Draw, 28
  - DrawShadow, 29
  - GetPositionVertices, 29
  - GetVertices, 29
  - Mesh, 28
  - shininess, 29
  - strength, 29
  - Update, 29
- minor\_version
  - ContextInfo, 10
- Model, 29
  - ~Model, 30
  - Destroy, 30
  - GetTexture, 30
  - GetVao, 31
  - GetVbos, 31
  - GetVertices, 31
  - Model, 30
  - SetProgram, 31
  - SetShadowProgram, 31
  - SetTexture, 31
  - textures, 32
  - transform, 32
  - vao, 32
  - vbos, 32
- Models\_Manager, 32
  - ~Models\_Manager, 33
  - addLight, 33
  - CreateModel, 33
  - CreateParticleSystem, 34
  - Draw, 34
  - DrawShadows, 34
  - Models\_Manager, 33
  - Update, 34
- msaa
  - FramebufferInfo, 13
- name
  - Texture, 47
  - WindowInfo, 54
- normal
  - VertexFormat, 53
- notifyBeginFrame
  - Scene\_Manager, 43
- notifyDisplayFrame
  - Scene\_Manager, 43
- notifyEndFrame
  - Scene\_Manager, 43
- notifyReshape
  - Scene\_Manager, 43
- operator<
  - VertexFormat, 52
- operator<<
  - VertexFormat, 52
- operator=
  - Camera, 9
  - ContextInfo, 10
  - Transform, 50
  - WindowInfo, 54
- Particle\_Container, 34
  - ~Particle\_Container, 35
  - Destroy, 35
  - Draw, 35
  - DrawShadow, 36
  - GetTexture, 36
  - Particle\_Container, 35
  - setMaxLife, 36
  - setMaxParticles, 36
  - setSpawnRate, 36
  - SetTexture, 36
  - transform, 37
  - Update, 37
- Physics\_Manager, 37
  - ~Physics\_Manager, 37
  - AddConstraint, 37
  - AddRigidBody, 37
  - DrawDebug, 37
  - GetInstance, 37
  - Step, 37
- PointLight, 38
  - Draw, 38

- PointLight, 38
- position
  - DVertexFormat, 13
  - Light, 25
  - Light::LightUniformLocations, 26
  - Transform, 51
  - VertexFormat, 53
- position\_x
  - WindowInfo, 54
- position\_y
  - WindowInfo, 54
- printOpenGLInfo
  - Init\_GLUT, 20
- program
  - IGameObject, 19
- quadraticAttenuation
  - Light, 25
  - Light::LightUniformLocations, 26
- render
  - DebugDrawer, 11
- RenderPass
  - Scene\_Manager, 43
- reportErrorWarning
  - DebugDrawer, 11
- resizeWindow
  - Camera, 9
- RotateX
  - Transform, 50
- RotateY
  - Transform, 50
- RotateZ
  - Transform, 50
- rotation
  - Transform, 51
- run
  - Init\_GLUT, 20
- scale
  - Transform, 51
- Scene\_Container, 39
  - ~Scene\_Container, 40
  - Destroy, 40
  - Draw, 40
  - DrawShadow, 40
  - GetMeshes, 40
  - getRigidBody, 40
  - GetTexture, 40
  - InitRigidBody, 41
  - Scene\_Container, 40
  - SetProgram, 41
  - SetShadowProgram, 41
  - SetTexture, 42
  - transform, 42
  - Update, 42
- Scene\_Manager, 42
  - ~Scene\_Manager, 43
  - GetDeltaTime, 43
  - GetFPS, 43
  - notifyBeginFrame, 43
  - notifyDisplayFrame, 43
  - notifyEndFrame, 43
  - notifyReshape, 43
  - RenderPass, 43
  - Scene\_Manager, 43
  - SetupScene, 43
  - ShadowPass, 43
  - UpdatePass, 43
- SetAttenuation
  - Light, 23
- setAttenuation
  - GameObjectsBuilder, 15
- setCastsShadows
  - GameObjectsBuilder, 15
- setColor
  - GameObjectsBuilder, 15
- setConeDirection
  - GameObjectsBuilder, 15
- setDebugMode
  - DebugDrawer, 11
- SetDepthMatrix
  - Shadow\_Manager, 45
- setDiffuse
  - GameObjectsBuilder, 15
- setHalfVector
  - GameObjectsBuilder, 15
- setListener
  - Init\_GLUT, 20
- setMaxLife
  - Particle\_Container, 36
- setMaxParticles
  - Particle\_Container, 36
- setNormal
  - GameObjectsBuilder, 15
- setParticleCount
  - GameObjectsBuilder, 15
- setParticleLife
  - GameObjectsBuilder, 15
- setParticleSpawnRate
  - GameObjectsBuilder, 15
- SetPosition
  - Transform, 50
- setPosition
  - GameObjectsBuilder, 15
- SetProgram
  - IGameObject, 17
  - Model, 31
  - Scene\_Container, 41
- SetRotation
  - Transform, 51
- setRotation
  - GameObjectsBuilder, 15
- SetScale
  - Transform, 51
- setScale
  - GameObjectsBuilder, 15

- SetShadowProgram
  - IGameObject, 18
  - Model, 31
  - Scene\_Container, 41
- setSpawnRate
  - Particle\_Container, 36
- setSpecular
  - GameObjectsBuilder, 15
- setSpotCutoff
  - GameObjectsBuilder, 15
- setSpotExponent
  - GameObjectsBuilder, 15
- setStrength
  - GameObjectsBuilder, 15
- SetTexture
  - IGameObject, 18
  - Light, 24
  - Model, 31
  - Particle\_Container, 36
  - Scene\_Container, 42
- SetTextureShader
  - Shader\_Factory, 44
- SetupScene
  - Scene\_Manager, 43
- Shader\_Factory, 43
  - ~Shader\_Factory, 44
  - CreateDebugProgram, 44
  - CreateProgram, 44
  - GetInstance, 44
  - SetTextureShader, 44
- Shadow\_Manager, 44
  - BindForWriting, 45
  - DEPTH\_TEXTURE\_SIZE, 46
  - GetDepthMatrix, 45
  - GetInstance, 45
  - GetShadowMap, 45
  - SetDepthMatrix, 45
  - Unbind, 46
- ShadowPass
  - Scene\_Manager, 43
- shadowProgram
  - IGameObject, 19
- shininess
  - Mesh, 29
- spotCosCutoff
  - Light, 25
  - Light::LightUniformLocations, 26
- spotExponent
  - Light, 25
  - Light::LightUniformLocations, 26
- SpotLight, 46
  - Draw, 47
  - SpotLight, 47
- Step
  - Physics\_Manager, 37
- strength
  - Mesh, 29
- tangent
  - VertexFormat, 53
- Texture, 47
  - id, 47
  - name, 47
  - type, 47
- texture
  - Light, 25
- textures
  - Model, 32
- Transform, 47
  - ~Transform, 49
  - getRotationMatrix, 49
  - getScaleMatrix, 49
  - getTransformMatrix, 49
  - getTranslationMatrix, 49
  - IncRotation, 49
  - operator=, 50
  - position, 51
  - RotateX, 50
  - RotateY, 50
  - RotateZ, 50
  - rotation, 51
  - scale, 51
  - SetPosition, 50
  - SetRotation, 51
  - SetScale, 51
  - Transform, 49
- transform
  - Model, 32
  - Particle\_Container, 37
  - Scene\_Container, 42
- type
  - Light, 25
  - Light::LightUniformLocations, 26
  - Texture, 47
- Unbind
  - Shadow\_Manager, 46
- Update
  - IGameObject, 18
  - Light, 24
  - Mesh, 29
  - Models\_Manager, 34
  - Particle\_Container, 37
  - Scene\_Container, 42
- UpdatePass
  - Scene\_Manager, 43
- uv
  - VertexFormat, 53
- vao
  - Model, 32
- VboIndexer, 51
  - indexVBO, 52
- vbos
  - Model, 32
- VertexFormat, 52
  - bitangent, 53
  - normal, 53

- operator<, 52
- operator<<, 52
- position, 53
- tangent, 53
- uv, 53
- VertexFormat, 52

width

- WindowInfo, 54

WindowInfo, 53

- height, 54
- isReshapable, 54
- name, 54
- operator=, 54
- position\_x, 54
- position\_y, 54
- width, 54
- WindowInfo, 54

zFar

- Camera, 10

zNear

- Camera, 10