

# Roombots Simulator

1.0

Generated by Doxygen 1.8.11



# Contents

<b>1</b>	<b>test</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Installation . . . . .	1
1.2.1	Step 1: Opening the box . . . . .	1
<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>File Index</b>	<b>7</b>
4.1	File List . . . . .	7
<b>5</b>	<b>Class Documentation</b>	<b>9</b>
5.1	BrutePathFinder Class Reference . . . . .	9
5.1.1	Detailed Description . . . . .	10
5.1.2	Member Function Documentation . . . . .	10
5.1.2.1	Name() const . . . . .	10
5.1.2.2	Run(Path &path, const Position &start, const Position &finish) const . . . . .	10
5.2	Button Class Reference . . . . .	11
5.2.1	Detailed Description . . . . .	11
5.2.2	Constructor & Destructor Documentation . . . . .	12
5.2.2.1	Button(glm::vec3 position, unsigned int ID, Structure *p_structure) . . . . .	12
5.2.3	Member Function Documentation . . . . .	12

5.2.3.1	AssignedStructure() const . . . . .	12
5.2.3.2	CleanUp() const . . . . .	12
5.2.3.3	Draw(const glm::mat4 &VP) const . . . . .	12
5.2.3.4	ID() const . . . . .	12
5.2.3.5	Position() const . . . . .	12
5.2.4	Member Data Documentation . . . . .	12
5.2.4.1	d_ID . . . . .	12
5.2.4.2	d_model . . . . .	12
5.2.4.3	d_p_structure . . . . .	13
5.2.4.4	d_position . . . . .	13
5.2.4.5	d_shadow . . . . .	13
5.3	Cube Class Reference . . . . .	13
5.3.1	Constructor & Destructor Documentation . . . . .	14
5.3.1.1	Cube(const std::string vShaderFileName, const std::string fShaderFileName, const std::string textureFileName, const glm::vec4 &color) . . . . .	14
5.3.1.2	Cube(const char *vShaderFileName, const char *fShaderFileName, const char *textureFileName, const glm::vec4 &color) . . . . .	14
5.3.2	Member Function Documentation . . . . .	14
5.3.2.1	SetUVs(std::vector< glm::vec2 > *uvs) . . . . .	14
5.3.2.2	SetVertices(std::vector< glm::vec3 > *vertices) . . . . .	14
5.4	DepthBuffer Struct Reference . . . . .	14
5.4.1	Constructor & Destructor Documentation . . . . .	15
5.4.1.1	DepthBuffer(OVR::SizeI size) . . . . .	15
5.4.2	Member Data Documentation . . . . .	15
5.4.2.1	texId . . . . .	15
5.5	GUI Class Reference . . . . .	15
5.5.1	Detailed Description . . . . .	16
5.5.2	Member Function Documentation . . . . .	16
5.5.2.1	AddButton(Structure *p_structure) . . . . .	16
5.5.2.2	CheckForPinchedStructure() . . . . .	16
5.5.2.3	CleanUp() . . . . .	17

5.5.2.4	DroppedStructure(unsigned int buttonID)	17
5.5.2.5	GetAllRoombotsPositions()	17
5.5.2.6	Init()	17
5.5.2.7	NButtons()	17
5.5.2.8	PopStructure(unsigned int buttonID)	17
5.5.2.9	Render(const glm::mat4 &VP)	17
5.5.2.10	Update(bool mode)	17
5.5.2.11	UpdatePointer(bool mode)	18
5.5.2.12	UpdateWorldMatrix(const glm::mat4 &worldMatrix)	18
5.5.3	Member Data Documentation	18
5.5.3.1	d_buttons	18
5.5.3.2	d_nButtons	18
5.5.3.3	d_nStructures	18
5.5.3.4	d_p_circle	18
5.5.3.5	d_p_hemi1	18
5.5.3.6	d_p_hemi2	18
5.5.3.7	d_pointer	18
5.5.3.8	d_structures	19
5.5.3.9	d_trashCan	19
5.6	HalfModule Class Reference	19
5.6.1	Detailed Description	20
5.6.2	Constructor & Destructor Documentation	20
5.6.2.1	HalfModule(Position position, OBJModel *p_h1, OBJModel *p_h2, OBJModel *p_circle)	20
5.6.2.2	HalfModule(int, int, int, OBJModel *p_h1, OBJModel *p_h2, OBJModel *p_circle)	20
5.6.3	Member Function Documentation	20
5.6.3.1	CleanUp()	20
5.6.3.2	Draw(const glm::mat4 &VP) const	20
5.6.3.3	GetPosition() const	20
5.6.3.4	SetPosition(const Position &position)	21
5.6.4	Member Data Documentation	21

5.6.4.1	<a href="#">d_p_circle</a>	21
5.6.4.2	<a href="#">d_p_hemisphere1</a>	21
5.6.4.3	<a href="#">d_p_hemisphere2</a>	21
5.6.4.4	<a href="#">d_position</a>	21
5.7	<a href="#">LeapmotionPointer Class Reference</a>	21
5.7.1	<a href="#">Detailed Description</a>	22
5.7.2	<a href="#">Member Function Documentation</a>	22
5.7.2.1	<a href="#">AdaptToMode(Leap::Vector right_hand_pos, bool mode)</a>	22
5.7.2.2	<a href="#">AssignedStructure() const</a>	23
5.7.2.3	<a href="#">AssignStructure(MovableStructure *p_structure)</a>	23
5.7.2.4	<a href="#">CleanUp()</a>	23
5.7.2.5	<a href="#">Draw(const glm::mat4 &amp;VP) const</a>	23
5.7.2.6	<a href="#">Init(GUI *p_gui)</a>	23
5.7.2.7	<a href="#">Pinching() const</a>	23
5.7.2.8	<a href="#">Position() const</a>	23
5.7.2.9	<a href="#">update(bool mode)</a>	23
5.7.2.10	<a href="#">UpdateWorldMatrix(const glm::mat4 &amp;worldMatrix)</a>	24
5.7.3	<a href="#">Member Data Documentation</a>	24
5.7.3.1	<a href="#">d_controller</a>	24
5.7.3.2	<a href="#">d_init</a>	24
5.7.3.3	<a href="#">d_invertedWorldMatrix</a>	24
5.7.3.4	<a href="#">d_p_gui</a>	24
5.7.3.5	<a href="#">d_p_pointerModel</a>	24
5.7.3.6	<a href="#">d_p_referencePointerModel</a>	24
5.7.3.7	<a href="#">d_p_shadow</a>	24
5.7.3.8	<a href="#">d_p_structure</a>	24
5.7.3.9	<a href="#">d_position</a>	25
5.8	<a href="#">MovableStructure Class Reference</a>	25
5.8.1	<a href="#">Detailed Description</a>	26
5.8.2	<a href="#">Constructor &amp; Destructor Documentation</a>	26

5.8.2.1	MovableStructure(Structure *p_structure, glm::vec3 position, int ID, unsigned int buttonID) . . . . .	26
5.8.3	Member Function Documentation . . . . .	26
5.8.3.1	CloseEnough(glm::vec3 position) const . . . . .	26
5.8.3.2	CloseEnough(glm::vec3 position, float distance) const . . . . .	26
5.8.3.3	Drag(const glm::vec3 &position) . . . . .	27
5.8.3.4	Draw(const glm::mat4 &VP) const . . . . .	27
5.8.3.5	Drop() . . . . .	27
5.8.3.6	GetPosition() const . . . . .	27
5.8.3.7	LinkedButtonID() const . . . . .	27
5.8.3.8	RoombotsPositions() const . . . . .	27
5.8.3.9	SetCenterOffset() . . . . .	27
5.8.3.10	SetPosition(glm::vec3 position) . . . . .	27
5.8.4	Member Data Documentation . . . . .	27
5.8.4.1	d_buttonID . . . . .	27
5.8.4.2	d_ID . . . . .	28
5.8.4.3	d_moving . . . . .	28
5.8.4.4	d_p_structure . . . . .	28
5.8.4.5	d_position . . . . .	28
5.9	OBJModel Class Reference . . . . .	28
5.9.1	Detailed Description . . . . .	29
5.9.2	Constructor & Destructor Documentation . . . . .	29
5.9.2.1	OBJModel(const std::string OBJFilename, const char *vShaderFileName, const char *fShaderFileName, const char *textureFileName, const glm::vec4 &color) . . . . .	29
5.9.3	Member Function Documentation . . . . .	30
5.9.3.1	SetUVs(std::vector< glm::vec2 > *uvs) . . . . .	30
5.9.3.2	SetVertices(std::vector< glm::vec3 > *vertices) . . . . .	30
5.9.4	Member Data Documentation . . . . .	30
5.9.4.1	_objfilename . . . . .	30
5.10	PathFinder Class Reference . . . . .	30
5.10.1	Detailed Description . . . . .	31

5.10.2	Member Function Documentation	31
5.10.2.1	Name() const =0	31
5.10.2.2	Run(Path &path, const Position &start, const Position &finish) const =0	31
5.11	Position Class Reference	31
5.11.1	Detailed Description	32
5.11.2	Constructor & Destructor Documentation	32
5.11.2.1	Position()	32
5.11.2.2	Position(int x, int y, int z)	32
5.11.2.3	Position(Position *)	32
5.11.2.4	Position(glm::vec3)	32
5.11.3	Member Function Documentation	32
5.11.3.1	distanceTo(Position other) const	32
5.11.3.2	operator!=(Position other) const	33
5.11.3.3	operator*(int factor) const	33
5.11.3.4	operator*=(int factor)	33
5.11.3.5	operator+(Position other) const	33
5.11.3.6	operator+=(Position other)	33
5.11.3.7	operator-(Position other) const	33
5.11.3.8	operator-=(Position other)	33
5.11.3.9	operator==(Position other) const	33
5.11.3.10	Print() const	33
5.11.3.11	ToGLM() const	33
5.11.3.12	x() const	33
5.11.3.13	y() const	33
5.11.3.14	z() const	33
5.11.4	Member Data Documentation	33
5.11.4.1	d_x	33
5.11.4.2	d_y	33
5.11.4.3	d_z	33
5.12	Quad Class Reference	34



5.12.1	Constructor & Destructor Documentation . . . . .	35
5.12.1.1	Quad(const std::string vShaderFileName, const std::string fShaderFileName, const std::string textureFileName, const glm::vec4 &color) . . . . .	35
5.12.1.2	Quad(const char *vShaderFileName, const char *fShaderFileName, const char *textureFileName, const glm::vec4 &color) . . . . .	35
5.12.2	Member Function Documentation . . . . .	35
5.12.2.1	SetUVs(std::vector< glm::vec2 > *uvs) . . . . .	35
5.12.2.2	SetVertices(std::vector< glm::vec3 > *vertices) . . . . .	35
5.13	RiftHandler Class Reference . . . . .	35
5.13.1	Member Function Documentation . . . . .	36
5.13.1.1	CleanUp() . . . . .	36
5.13.1.2	DisplayOnRift() . . . . .	36
5.13.1.3	glmViewProjMatrix() . . . . .	36
5.13.1.4	Init(DisplayFunction) . . . . .	36
5.13.1.5	ovrViewProjMatrix() . . . . .	37
5.13.1.6	ResolutionHeight() . . . . .	37
5.13.1.7	ResolutionWidth() . . . . .	37
5.13.2	Member Data Documentation . . . . .	37
5.13.2.1	d_displayFunction . . . . .	37
5.13.2.2	d_eyeDepthBuffer . . . . .	37
5.13.2.3	d_EyeRenderDesc . . . . .	37
5.13.2.4	d_eyeRenderTexture . . . . .	37
5.13.2.5	d_hmd . . . . .	37
5.13.2.6	d_isVisible . . . . .	37
5.13.2.7	d_mirrorFBO . . . . .	37
5.13.2.8	d_mirrorTexture . . . . .	38
5.13.2.9	d_trackingState . . . . .	38
5.13.2.10	d_viewProjMatrix . . . . .	38
5.14	RoomBot Class Reference . . . . .	38
5.14.1	Detailed Description . . . . .	39
5.14.2	Constructor & Destructor Documentation . . . . .	39

5.14.2.1	RoomBot(Position A, Position B, OBJModel *p_h1, OBJModel *p_h2, OBJModel *p_circle) . . . . .	39
5.14.2.2	RoomBot(int Ax, int Ay, int Az, int Bx, int By, int Bz, OBJModel *p_h1, OBJModel *p_h2, OBJModel *p_circle) . . . . .	39
5.14.3	Member Function Documentation . . . . .	39
5.14.3.1	Draw(const glm::mat4 &VP) const . . . . .	39
5.14.3.2	MiddlePosition() const . . . . .	40
5.14.3.3	PositionA() const . . . . .	40
5.14.3.4	PositionB() const . . . . .	40
5.14.4	Member Data Documentation . . . . .	40
5.14.4.1	d_halfModuleA . . . . .	40
5.14.4.2	d_halfModuleB . . . . .	40
5.15	Scene Class Reference . . . . .	40
5.15.1	Detailed Description . . . . .	41
5.15.2	Member Function Documentation . . . . .	41
5.15.2.1	AddModel(Model *sourceModel) . . . . .	41
5.15.2.2	CleanUp() . . . . .	41
5.15.2.3	Init(float roomSize) . . . . .	41
5.15.2.4	Render(const glm::mat4 &VP, bool drawRoof) . . . . .	42
5.15.3	Member Data Documentation . . . . .	42
5.15.3.1	d_models . . . . .	42
5.15.3.2	d_nModels . . . . .	42
5.15.3.3	d_roof . . . . .	42
5.16	ShaderLoader Class Reference . . . . .	42
5.16.1	Member Function Documentation . . . . .	43
5.16.1.1	CreateProgram(const char *VertexShaderFilename, const char *FragmentShaderFilename) . . . . .	43
5.16.1.2	CreateShader(GLenum shaderType, std::string source, char *shaderName) . . . . .	43
5.16.1.3	DefaultFragmentShader() . . . . .	43
5.16.1.4	DefaultVertexShader() . . . . .	43
5.16.1.5	ReadShader(const char *filename) . . . . .	44
5.17	Simulation Class Reference . . . . .	44

5.17.1 Detailed Description . . . . .	45
5.17.2 Member Function Documentation . . . . .	45
5.17.2.1 Draw(const glm::mat4 &VP) . . . . .	45
5.17.2.2 Initialize(const std::vector< Position > roombotsFinalPositions, Pathfinder *pathFinder) . . . . .	45
5.17.2.3 IsInitialized() . . . . .	45
5.17.2.4 IsOver() . . . . .	45
5.17.2.5 NextStep() . . . . .	45
5.17.2.6 Reset() . . . . .	45
5.17.2.7 Run() . . . . .	45
5.17.3 Member Data Documentation . . . . .	46
5.17.3.1 d_currentStep . . . . .	46
5.17.3.2 d_halfModules . . . . .	46
5.17.3.3 d_init . . . . .	46
5.17.3.4 d_over . . . . .	46
5.17.3.5 d_paths . . . . .	46
5.17.3.6 d_refClock . . . . .	46
5.18 Simulator Class Reference . . . . .	46
5.18.1 Detailed Description . . . . .	48
5.18.2 Constructor & Destructor Documentation . . . . .	48
5.18.2.1 Simulator() . . . . .	48
5.18.2.2 ~Simulator() . . . . .	48
5.18.3 Member Function Documentation . . . . .	48
5.18.3.1 Backwards() . . . . .	48
5.18.3.2 CleanUp() . . . . .	48
5.18.3.3 Close() . . . . .	48
5.18.3.4 Display() . . . . .	48
5.18.3.5 Forward() . . . . .	48
5.18.3.6 HandleKeyboard(unsigned char key, int x, int y) . . . . .	48
5.18.3.7 Init(int argc, char **argv, DisplayFunction display, DisplayFunction render↔ Scene, void(*keyboardFunc)(unsigned char, int, int), void(*resizeFunc)(int, int), void(*closeFunc)()) . . . . .	49

5.18.3.8	InitRift(DisplayFunction function)	49
5.18.3.9	InitSimulation()	49
5.18.3.10	Instance()	49
5.18.3.11	Left()	49
5.18.3.12	MainLoop()	49
5.18.3.13	RenderScene()	49
5.18.3.14	Resize(int w, int h)	49
5.18.3.15	Right()	49
5.18.3.16	Start()	49
5.18.3.17	SwitchViewMode()	50
5.18.3.18	WorldViewMatrix()	50
5.18.4	Member Data Documentation	50
5.18.4.1	d_GUI	50
5.18.4.2	d_height	50
5.18.4.3	d_instance	50
5.18.4.4	d_mode	50
5.18.4.5	d_pathFinder	50
5.18.4.6	d_rift	50
5.18.4.7	d_running	50
5.18.4.8	d_scene	50
5.18.4.9	d_simulation	51
5.18.4.10	d_width	51
5.18.4.11	d_windowID	51
5.18.4.12	d_worldMatrix	51
5.19	Structure Class Reference	51
5.19.1	Detailed Description	52
5.19.2	Constructor & Destructor Documentation	52
5.19.2.1	Structure(std::string sourceFilename, OBJModel *p_h1, OBJModel *p_h2, OBJModel *p_circle)	52
5.19.3	Member Function Documentation	52
5.19.3.1	CenterOffset() const	52

5.19.3.2	Draw(const glm::mat4 &VP) const	52
5.19.3.3	RoombotsPositions() const	52
5.19.3.4	SetCenterOffset()	52
5.19.4	Member Data Documentation	53
5.19.4.1	d_centerOffset	53
5.19.4.2	d_filename	53
5.19.4.3	d_roomBots	53
5.20	TextureBuffer Struct Reference	53
5.20.1	Constructor & Destructor Documentation	54
5.20.1.1	TextureBuffer(ovrHmd hmd, bool rendertarget, bool displayableOnHmd, OVR::Sizei size, int mipLevels, unsigned char *data, int sampleCount)	54
5.20.1.2	TextureBuffer(ovrHmd hmd, bool rendertarget, bool displayableOnHmd, OVR::Sizei size, int mipLevels, unsigned char *data, int sampleCount)	54
5.20.2	Member Function Documentation	54
5.20.2.1	GetSize(void) const	54
5.20.2.2	GetSize(void) const	54
5.20.2.3	SetAndClearRenderSurface(DepthBuffer *dbuffer)	54
5.20.2.4	SetAndClearRenderSurface(DepthBuffer *dbuffer)	54
5.20.2.5	UnsetRenderSurface()	54
5.20.2.6	UnsetRenderSurface()	54
5.20.3	Member Data Documentation	54
5.20.3.1	fbold	54
5.20.3.2	texId	54
5.20.3.3	texSize	54
5.20.3.4	TextureSet	54
5.21	TrashCan Class Reference	55
5.21.1	Detailed Description	55
5.21.2	Constructor & Destructor Documentation	55
5.21.2.1	TrashCan(glm::vec3 position)	55
5.21.3	Member Function Documentation	56
5.21.3.1	CleanUp() const	56
5.21.3.2	Draw(const glm::mat4 &VP) const	56
5.21.3.3	Position() const	56
5.21.4	Member Data Documentation	56
5.21.4.1	d_model	56
5.21.4.2	d_position	56

<b>6</b>	<b>File Documentation</b>	<b>57</b>
6.1	RoombotsSimulator/BrutePathFinder.cc File Reference	57
6.2	RoombotsSimulator/BrutePathFinder.hh File Reference	57
6.3	RoombotsSimulator/Button.cc File Reference	58
6.4	RoombotsSimulator/Button.hh File Reference	58
6.5	RoombotsSimulator/common.hh File Reference	58
6.5.1	Macro Definition Documentation	60
6.5.1.1	BOX_COORDINATE_SYSTEM_SCALE_CONVERSION	60
6.5.1.2	BUTTON_DEPTH_OFFSET	60
6.5.1.3	BUTTON_RIGHT_START	60
6.5.1.4	BUTTON_SEPARATION	60
6.5.1.5	BUTTON_SIZE	60
6.5.1.6	BUTTON_UP_START	60
6.5.1.7	COORDINATE_SYSTEM_SCALE_CONVERSION	60
6.5.1.8	DRAG_RADIUS	61
6.5.1.9	EYES_POSITION	61
6.5.1.10	LEAP_POINTER_SIZE	61
6.5.1.11	MODULE_SIZE	61
6.5.1.12	PINCHING_LIMIT	61
6.5.1.13	ROOM_SIZE	61
6.5.1.14	TRASH_CAN_SIZE	61
6.6	RoombotsSimulator/Cube.cc File Reference	61
6.7	RoombotsSimulator/Cube.hh File Reference	62
6.8	RoombotsSimulator/DepthBuffer.cc File Reference	62
6.9	RoombotsSimulator/DepthBuffer.hh File Reference	62
6.10	RoombotsSimulator/GUI.cc File Reference	63
6.11	RoombotsSimulator/GUI.hh File Reference	63
6.12	RoombotsSimulator/HalfModule.cc File Reference	63
6.13	RoombotsSimulator/HalfModule.hh File Reference	64
6.14	RoombotsSimulator/LeapmotionPointer.cc File Reference	64

6.15 RoombotsSimulator/LeapmotionPointer.hh File Reference . . . . .	64
6.16 RoombotsSimulator/main.cc File Reference . . . . .	65
6.16.1 Function Documentation . . . . .	65
6.16.1.1 close() . . . . .	65
6.16.1.2 display() . . . . .	65
6.16.1.3 handleKeyboard(unsigned char key, int x, int y) . . . . .	66
6.16.1.4 main(int argc, char **argv) . . . . .	66
6.16.1.5 renderScene() . . . . .	66
6.16.1.6 resize(int w, int h) . . . . .	66
6.17 RoombotsSimulator/mainpage.dox File Reference . . . . .	66
6.18 RoombotsSimulator/Model.cc File Reference . . . . .	66
6.19 RoombotsSimulator/Model.hh File Reference . . . . .	66
6.19.1 Function Documentation . . . . .	67
6.19.1.1 __declspec(align(16)) class Model . . . . .	67
6.20 RoombotsSimulator/MovableStructure.cc File Reference . . . . .	68
6.21 RoombotsSimulator/MovableStructure.hh File Reference . . . . .	68
6.22 RoombotsSimulator/OBJModel.cc File Reference . . . . .	68
6.23 RoombotsSimulator/OBJModel.hh File Reference . . . . .	68
6.24 RoombotsSimulator/PathFinder.hh File Reference . . . . .	69
6.24.1 Typedef Documentation . . . . .	70
6.24.1.1 Path . . . . .	70
6.25 RoombotsSimulator/Position.cc File Reference . . . . .	70
6.26 RoombotsSimulator/Position.hh File Reference . . . . .	70
6.27 RoombotsSimulator/Quad.cc File Reference . . . . .	71
6.28 RoombotsSimulator/Quad.hh File Reference . . . . .	71
6.29 RoombotsSimulator/RiftHandler.cc File Reference . . . . .	71
6.29.1 Function Documentation . . . . .	72
6.29.1.1 OVR_Mat4_to_GLM_mat4(OVR::Matrix4f sourceMatrix) . . . . .	72
6.30 RoombotsSimulator/RiftHandler.hh File Reference . . . . .	72
6.30.1 Typedef Documentation . . . . .	72

6.30.1.1 DisplayFunction . . . . .	72
6.31 RoombotsSimulator/RoomBot.cc File Reference . . . . .	72
6.32 RoombotsSimulator/RoomBot.hh File Reference . . . . .	73
6.33 RoombotsSimulator/Scene.cc File Reference . . . . .	73
6.34 RoombotsSimulator/Scene.hh File Reference . . . . .	73
6.35 RoombotsSimulator/ShaderLoader.cc File Reference . . . . .	74
6.36 RoombotsSimulator/ShaderLoader.hh File Reference . . . . .	74
6.37 RoombotsSimulator/Simulation.cc File Reference . . . . .	74
6.38 RoombotsSimulator/Simulation.hh File Reference . . . . .	75
6.39 RoombotsSimulator/Simulator.cc File Reference . . . . .	75
6.40 RoombotsSimulator/Simulator.hh File Reference . . . . .	75
6.41 RoombotsSimulator/Structure.cc File Reference . . . . .	76
6.41.1 Function Documentation . . . . .	76
6.41.1.1 min(int a, int b) . . . . .	76
6.41.1.2 min(int a, int b, int c) . . . . .	76
6.42 RoombotsSimulator/Structure.hh File Reference . . . . .	76
6.43 RoombotsSimulator/TextureBuffer.cc File Reference . . . . .	77
6.44 RoombotsSimulator/TextureBuffer.hh File Reference . . . . .	77
6.45 RoombotsSimulator/TrashCan.cc File Reference . . . . .	77
6.46 RoombotsSimulator/TrashCan.hh File Reference . . . . .	78
<b>Index</b>	<b>79</b>



# Chapter 1

## test

### 1.1 Introduction

This is the introduction.

### 1.2 Installation

#### 1.2.1 Step 1: Opening the box



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

Button . . . . .	11
DepthBuffer . . . . .	14
GUI . . . . .	15
HalfModule . . . . .	19
LeapmotionPointer . . . . .	21
Model	
Cube . . . . .	13
OBJModel . . . . .	28
Quad . . . . .	34
MovableStructure . . . . .	25
PathFinder . . . . .	30
BrutePathFinder . . . . .	9
Position . . . . .	31
RiftHandler . . . . .	35
RoomBot . . . . .	38
Scene . . . . .	40
ShaderLoader . . . . .	42
Simulation . . . . .	44
Simulator . . . . .	46
Structure . . . . .	51
TextureBuffer . . . . .	53
TrashCan . . . . .	55



## Chapter 3

# Class Index

### 3.1 Class List

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

BrutePathFinder	9
Button	11
Cube	13
DepthBuffer	14
GUI	15
HalfModule	19
LeapmotionPointer	21
MovableStructure	25
OBJModel	28
PathFinder	30
Position	31
Quad	34
RiftHandler	35
RoomBot	38
Scene	40
ShaderLoader	42
Simulation	44
Simulator	46
Structure	51
TextureBuffer	53
TrashCan	55



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

RoombotsSimulator/BrutePathFinder.cc	57
RoombotsSimulator/BrutePathFinder.hh	57
RoombotsSimulator/Button.cc	58
RoombotsSimulator/Button.hh	58
RoombotsSimulator/common.hh	58
RoombotsSimulator/Cube.cc	61
RoombotsSimulator/Cube.hh	62
RoombotsSimulator/DepthBuffer.cc	62
RoombotsSimulator/DepthBuffer.hh	62
RoombotsSimulator/GUI.cc	63
RoombotsSimulator/GUI.hh	63
RoombotsSimulator/HalfModule.cc	63
RoombotsSimulator/HalfModule.hh	64
RoombotsSimulator/LeapmotionPointer.cc	64
RoombotsSimulator/LeapmotionPointer.hh	64
RoombotsSimulator/main.cc	65
RoombotsSimulator/Model.cc	66
RoombotsSimulator/Model.hh	66
RoombotsSimulator/MovableStructure.cc	68
RoombotsSimulator/MovableStructure.hh	68
RoombotsSimulator/OBJModel.cc	68
RoombotsSimulator/OBJModel.hh	68
RoombotsSimulator/PathFinder.hh	69
RoombotsSimulator/Position.cc	70
RoombotsSimulator/Position.hh	70
RoombotsSimulator/Quad.cc	71
RoombotsSimulator/Quad.hh	71
RoombotsSimulator/RiftHandler.cc	71
RoombotsSimulator/RiftHandler.hh	72
RoombotsSimulator/RoomBot.cc	72
RoombotsSimulator/RoomBot.hh	73
RoombotsSimulator/Scene.cc	73
RoombotsSimulator/Scene.hh	73
RoombotsSimulator/ShaderLoader.cc	74
RoombotsSimulator/ShaderLoader.hh	74

RoombotsSimulator/ <a href="#">Simulation.cc</a> . . . . .	74
RoombotsSimulator/ <a href="#">Simulation.hh</a> . . . . .	75
RoombotsSimulator/ <a href="#">Simulator.cc</a> . . . . .	75
RoombotsSimulator/ <a href="#">Simulator.hh</a> . . . . .	75
RoombotsSimulator/ <a href="#">Structure.cc</a> . . . . .	76
RoombotsSimulator/ <a href="#">Structure.hh</a> . . . . .	76
RoombotsSimulator/ <a href="#">TextureBuffer.cc</a> . . . . .	77
RoombotsSimulator/ <a href="#">TextureBuffer.hh</a> . . . . .	77
RoombotsSimulator/ <a href="#">TrashCan.cc</a> . . . . .	77
RoombotsSimulator/ <a href="#">TrashCan.hh</a> . . . . .	78



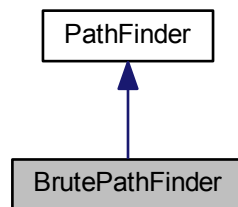
## Chapter 5

# Class Documentation

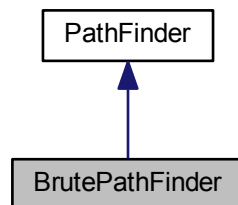
### 5.1 BrutePathFinder Class Reference

```
#include <BrutePathFinder.hh>
```

Inheritance diagram for BrutePathFinder:



Collaboration diagram for BrutePathFinder:



## Public Member Functions

- virtual void [Run](#) ([Path](#) &path, const [Position](#) &start, const [Position](#) &finish) const
- virtual std::string [Name](#) () const

### 5.1.1 Detailed Description

This [PathFinder](#) uses a very simple path-finding algorithm : It simply goes to the same z-coordinate than the final position, then to the same x-coordinate and finally to the same y-coordinate, to end up at the same [Position](#)

### 5.1.2 Member Function Documentation

#### 5.1.2.1 std::string BrutePathFinder::Name ( ) const [virtual]

This method simply returns the name of the path-finding algorithm

Implements [PathFinder](#).

#### 5.1.2.2 void BrutePathFinder::Run ( [Path](#) & path, const [Position](#) & start, const [Position](#) & finish ) const [virtual]

Creates a Path, a succession of [Position](#) from a [Position](#) to another. It uses an external Path and fills it

- `path` A reference to the Path to fill
- `start` The starting [Position](#) of the Path
- `finish` The finishing [Position](#) of the Path

Implements [PathFinder](#).

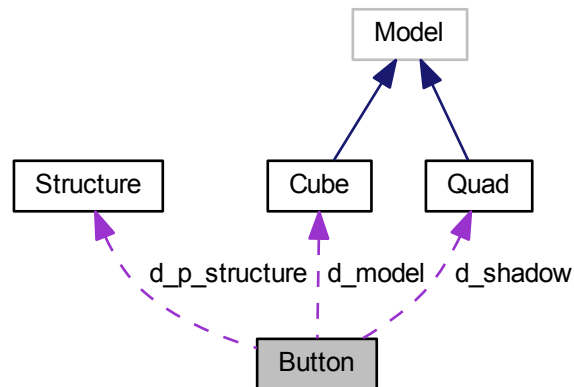
The documentation for this class was generated from the following files:

- RoombotsSimulator/[BrutePathFinder.hh](#)
- RoombotsSimulator/[BrutePathFinder.cc](#)

## 5.2 Button Class Reference

```
#include <Button.hh>
```

Collaboration diagram for Button:



### Public Member Functions

- [Button](#) (glm::vec3 position, unsigned int [ID](#), [Structure](#) \*p\_structure)
- void [Draw](#) (const glm::mat4 &VP) const
- void [CleanUp](#) () const
- glm::vec3 [Position](#) () const
- unsigned int [ID](#) () const
- [Structure](#) \* [AssignedStructure](#) () const

### Private Attributes

- [Cube](#) \* [d\\_model](#)  
*the cube model used to represent the button*
- [Quad](#) \* [d\\_shadow](#)  
*a shadow of the cube projected on the floor*
- const glm::vec3 [d\\_position](#)  
*its position within the scene*
- const unsigned int [d\\_ID](#)  
*its unique ID. Used to pop new Structures on the button*
- [Structure](#) \* [d\\_p\\_structure](#)  
*a pointer to the [Structure](#) that pops from the button*

### 5.2.1 Detailed Description

A [Button](#) is a holder for [MovableStructure](#) to pop.

## 5.2.2 Constructor & Destructor Documentation

### 5.2.2.1 `Button::Button ( glm::vec3 position, unsigned int ID, Structure * p_structure )`

Creates a new button containing the structure passed as argument

- `position` the position of the new [Button](#)
- `ID` the unique ID of the new [Button](#)
- a pointer to the [Structure](#) that will be used to pop new MovableStructures

## 5.2.3 Member Function Documentation

### 5.2.3.1 `Structure * Button::AssignedStructure ( ) const`

Returns a pointer to the assigned [Structure](#)

### 5.2.3.2 `void Button::CleanUp ( ) const`

Cleans up all the models (cube, shadow and the structure's) and deletes the pointer to the [Structure](#), meaning it absolutely shouldn't be accessed once this method has been called

### 5.2.3.3 `void Button::Draw ( const glm::mat4 & VP ) const`

Draws the button and its shadow

### 5.2.3.4 `unsigned int Button::ID ( ) const` `[inline]`

Returns the button's ID

### 5.2.3.5 `glm::vec3 Button::Position ( ) const` `[inline]`

Returns the button's position

## 5.2.4 Member Data Documentation

### 5.2.4.1 `const unsigned int Button::d_ID` `[private]`

its unique ID. Used to pop new Structures on the button

### 5.2.4.2 `Cube* Button::d_model` `[private]`

the cube model used to represent the button

**5.2.4.3** `Structure* Button::d_p_structure` `[private]`

a pointer to the [Structure](#) that pops from the button

**5.2.4.4** `const glm::vec3 Button::d_position` `[private]`

its position within the scene

**5.2.4.5** `Quad* Button::d_shadow` `[private]`

a shadow of the cube projected on the floor

The documentation for this class was generated from the following files:

- RoombotsSimulator/[Button.hh](#)
- RoombotsSimulator/[Button.cc](#)

## 5.3 Cube Class Reference

```
#include <Cube.hh>
```

Inheritance diagram for Cube:



Collaboration diagram for Cube:



## Public Member Functions

- [Cube](#) (const std::string vShaderFileName, const std::string fShaderFileName, const std::string textureFileName, const glm::vec4 &color)
- [Cube](#) (const char \*vShaderFileName, const char \*fShaderFileName, const char \*textureFileName, const glm::vec4 &color)

## Protected Member Functions

- virtual void [SetVertices](#) (std::vector< glm::vec3 > \*vertices)
- virtual void [SetUVs](#) (std::vector< glm::vec2 > \*uvs)

### 5.3.1 Constructor & Destructor Documentation

5.3.1.1 `Cube::Cube ( const std::string vShaderFileName, const std::string fShaderFileName, const std::string textureFileName, const glm::vec4 & color ) [inline]`

5.3.1.2 `Cube::Cube ( const char * vShaderFileName, const char * fShaderFileName, const char * textureFileName, const glm::vec4 & color ) [inline]`

### 5.3.2 Member Function Documentation

5.3.2.1 `void Cube::SetUVs ( std::vector< glm::vec2 > * uvs ) [protected], [virtual]`

5.3.2.2 `void Cube::SetVertices ( std::vector< glm::vec3 > * vertices ) [protected], [virtual]`

Creates and adds vertices to the Model to create a one-meter cube

The documentation for this class was generated from the following files:

- RoombotsSimulator/[Cube.hh](#)
- RoombotsSimulator/[Cube.cc](#)

## 5.4 DepthBuffer Struct Reference

```
#include <DepthBuffer.hh>
```

### Public Member Functions

- [DepthBuffer](#) (OVR::Sizei size)

### Public Attributes

- GLuint [texId](#)

### 5.4.1 Constructor & Destructor Documentation

5.4.1.1 `DepthBuffer::DepthBuffer ( OVR::Size size ) [inline]`

### 5.4.2 Member Data Documentation

5.4.2.1 `GLuint DepthBuffer::texId`

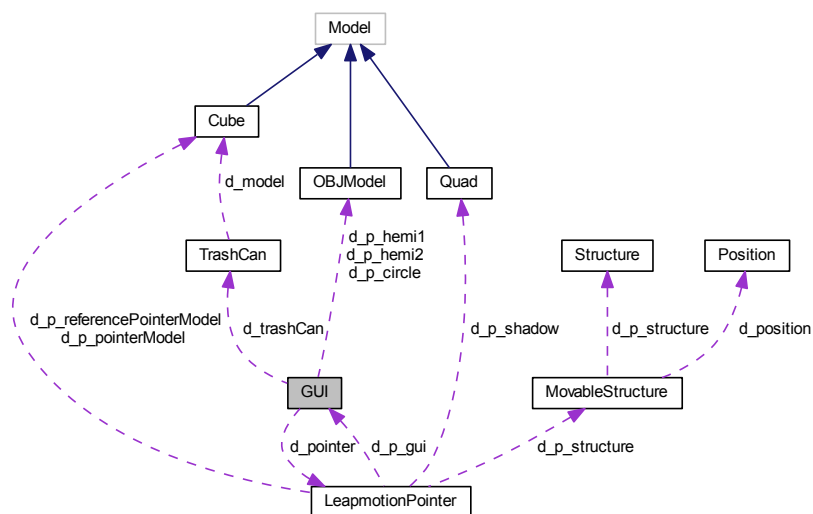
The documentation for this struct was generated from the following file:

- RoombotsSimulator/[DepthBuffer.hh](#)

## 5.5 GUI Class Reference

```
#include <GUI.hh>
```

Collaboration diagram for GUI:



### Public Member Functions

- void `Init` ()
- void `Update` (bool mode)
- void `UpdateWorldMatrix` (const glm::mat4 &worldMatrix)
- void const `Render` (const glm::mat4 &VP)
- size\_t `NButtons` ()
- void `DroppedStructure` (unsigned int buttonID)
- std::vector< `Position` > `GetAllRoombotsPositions` ()
- void `CleanUp` ()

## Private Member Functions

- void [AddButton](#) ([Structure](#) \*p\_structure)
- void [CheckForPinchedStructure](#) ()
- void [UpdatePointer](#) (bool mode)
- void [PopStructure](#) (unsigned int buttonID)

## Private Attributes

- std::vector< const [Button](#) \* > [d\\_buttons](#)  
*The Buttons contained in the scene.*
- size\_t [d\\_nButtons](#) = 0  
*The number of Buttons in the scene.*
- std::vector< [MovableStructure](#) \* > [d\\_structures](#)  
*The Structures contained in the scene.*
- size\_t [d\\_nStructures](#) = 0  
*The number of Structures in the scene.*
- [LeapmotionPointer](#) [d\\_pointer](#)  
*The pointer using the Leapmotion device.*
- [OBJModel](#) \* [d\\_p\\_hemi1](#)  
*The pointer to the first hemisphere Model.*
- [OBJModel](#) \* [d\\_p\\_hemi2](#)  
*The pointer to the second hemisphere Model.*
- [OBJModel](#) \* [d\\_p\\_circle](#)  
*The pointer to the circle Model.*
- [TrashCan](#) \* [d\\_trashCan](#)

### 5.5.1 Detailed Description

The Graphic User Interface ([GUI](#)) regroups everything related to the manipulation of the environment. It notably handles the [LeapmotionPointer](#), the Buttons, the [TrashCan](#) and all [MovableStructures](#). It is also the link between all those elements

### 5.5.2 Member Function Documentation

#### 5.5.2.1 void GUI::AddButton ( [Structure](#) \* p\_structure ) [private]

Adds a button to the interface (max 3 for now)

- p\_structure A pointer to the [Structure](#) that will be used to pop new [MovableStructures](#) from the added [Button](#)

#### 5.5.2.2 void GUI::CheckForPinchedStructure ( ) [private]

Checks for every [Structure](#) if it is being pinched by the [LeapmotionPointer](#)



### 5.5.2.3 void GUI::CleanUp ( )

Cleans up everything

### 5.5.2.4 void GUI::DroppedStructure ( unsigned int *buttonID* )

Notifies the GUI a Structure has been dropped and pops a new one on the corresponding button if the ID is valid

- *buttonID* The ID of the Button from which the dropped MovableStructure comes

### 5.5.2.5 std::vector< Position > GUI::GetAllRoombotsPositions ( )

Returns a vector of the positions of all the RoomBots

### 5.5.2.6 void GUI::Init ( )

Initializes the Graphic User Interface by initializing the LeapmotionPointer and adding Buttons

### 5.5.2.7 size\_t GUI::NButtons ( )

Returns the number of Buttons in the GUI

### 5.5.2.8 void GUI::PopStructure ( unsigned int *buttonID* ) [private]

Pops a new structure in a Button

- *buttonID* The ID of the Button where to pop a new MovableStructure

### 5.5.2.9 void const GUI::Render ( const glm::mat4 & *VP* )

Renders (draws) the Buttons, the Structures and the LeapmotionPointer

### 5.5.2.10 void GUI::Update ( bool *mode* )

Updates the positions of the Structures and the LeapmotionPointer

- *mode* The current Simulator mode.

#### 5.5.2.11 void GUI::UpdatePointer ( bool *mode* ) [private]

Updates the [LeapmotionPointer](#)'s position depending on the Leapmotion Controller's data

- *mode* the current [Simulator](#) mode

#### 5.5.2.12 void GUI::UpdateWorldMatrix ( const glm::mat4 & *worldMatrix* )

Updates the scene's world matrix. Used to change the behaviour of the [LeapmotionPointer](#) depending on the current mode

### 5.5.3 Member Data Documentation

#### 5.5.3.1 std::vector<const **Button**> GUI::d\_buttons [private]

The Buttons contained in the scene.

#### 5.5.3.2 size\_t GUI::d\_nButtons = 0 [private]

The number of Buttons in the scene.

#### 5.5.3.3 size\_t GUI::d\_nStructures = 0 [private]

The number of Structures in the scene.

#### 5.5.3.4 **OBJModel**\* GUI::d\_p\_circle [private]

The pointer to the circle Model.

#### 5.5.3.5 **OBJModel**\* GUI::d\_p\_hemi1 [private]

The pointer to the first hemisphere Model.

#### 5.5.3.6 **OBJModel**\* GUI::d\_p\_hemi2 [private]

The pointer to the second hemisphere Model.

#### 5.5.3.7 **LeapmotionPointer** GUI::d\_pointer [private]

The pointer using the Leapmotion device.

5.5.3.8 `std::vector<MovableStructure*> GUI::d_structures` [private]

The Structures contained in the scene.

5.5.3.9 `TrashCan* GUI::d_trashCan` [private]

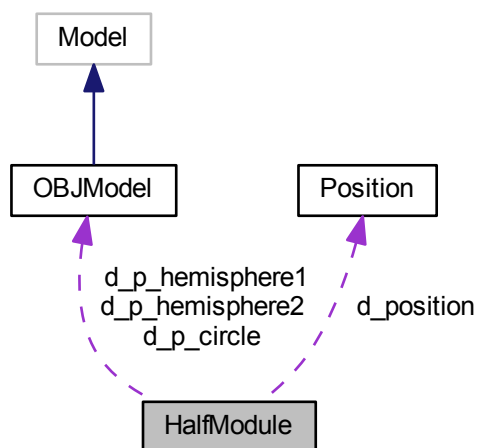
The documentation for this class was generated from the following files:

- RoombotsSimulator/[GUI.hh](#)
- RoombotsSimulator/[GUI.cc](#)

## 5.6 HalfModule Class Reference

```
#include <HalfModule.hh>
```

Collaboration diagram for HalfModule:



### Public Member Functions

- [HalfModule](#) ([Position](#) position, [OBJModel](#) \*p\_h1, [OBJModel](#) \*p\_h2, [OBJModel](#) \*p\_circle)
- [HalfModule](#) (int, int, int, [OBJModel](#) \*p\_h1, [OBJModel](#) \*p\_h2, [OBJModel](#) \*p\_circle)
- void [Draw](#) (const glm::mat4 &VP) const
- [Position](#) [GetPosition](#) () const
- void [SetPosition](#) (const [Position](#) &position)
- void [CleanUp](#) ()

## Private Attributes

- [Position](#) `d_position`  
The relative coordinates within the [Structure](#).
- [OBJModel](#) \* `d_p_hemisphere1`  
A pointer to a up-oriented hemisphere Model.
- [OBJModel](#) \* `d_p_hemisphere2`  
A pointer to a down-oriented hemisphere Model.
- [OBJModel](#) \* `d_p_circle`  
A pointer to a circle model.

### 5.6.1 Detailed Description

A [HalfModule](#) is the half of every [RoomBot](#) module

### 5.6.2 Constructor & Destructor Documentation

**5.6.2.1** `HalfModule::HalfModule ( Position position, OBJModel * p_h1, OBJModel * p_h2, OBJModel * p_circle )`

Creates a new [HalfModule](#)

- `position` The position of the new [HalfModule](#)
- `p_h1` A pointer to the first hemi-sphere [OBJModel](#) used to draw the [HalfModule](#)
- `p_h2` A pointer to the second hemi-sphere [OBJModel](#) used to draw the [HalfModule](#)
- `p_circle` A pointer to the circle [OBJModel](#) used to draw all six faces of the [HalfModule](#)

**5.6.2.2** `HalfModule::HalfModule ( int x, int y, int z, OBJModel * p_h1, OBJModel * p_h2, OBJModel * p_circle )`

Same as the first constructor, but with all three `int` used to create a new [Position](#)

### 5.6.3 Member Function Documentation

**5.6.3.1** `void HalfModule::CleanUp ( )`

Cleans up the three Models used in the [HalfModule](#)

**5.6.3.2** `void HalfModule::Draw ( const glm::mat4 & VP ) const`

Draws the two hemispheres and the circle six times in different positions and orientations.

**5.6.3.3** `Position HalfModule::GetPosition ( ) const`

Returns the relative [Position](#) of the halfModule within its [Structure](#)

#### 5.6.3.4 void HalfModule::SetPosition ( const Position & position )

Directly sets the position of the Module. This should only be used when running the simulation or when the Module is free from any [Structure](#)

### 5.6.4 Member Data Documentation

#### 5.6.4.1 OBJModel\* HalfModule::d\_p\_circle [private]

A pointer to a circle model.

#### 5.6.4.2 OBJModel\* HalfModule::d\_p\_hemisphere1 [private]

A pointer to a up-oriented hemisphere Model.

#### 5.6.4.3 OBJModel\* HalfModule::d\_p\_hemisphere2 [private]

A pointer to a down-oriented hemisphere Model.

#### 5.6.4.4 Position HalfModule::d\_position [private]

The relative coordinates within the [Structure](#).

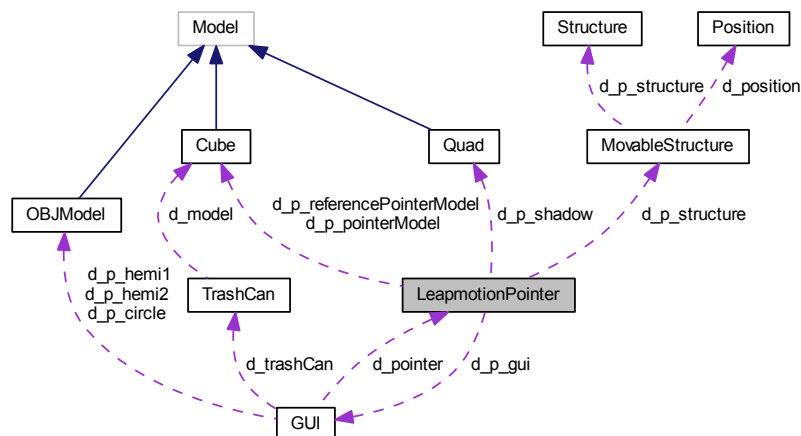
The documentation for this class was generated from the following files:

- [RoombotsSimulator/HalfModule.hh](#)
- [RoombotsSimulator/HalfModule.cc](#)

## 5.7 LeapmotionPointer Class Reference

```
#include <LeapmotionPointer.hh>
```

Collaboration diagram for LeapmotionPointer:



## Public Member Functions

- void [Init](#) ([GUI](#) \*p\_gui)
- void [update](#) (bool mode)
- void [UpdateWorldMatrix](#) (const glm::mat4 &worldMatrix)
- void [Draw](#) (const glm::mat4 &VP) const
- bool [Pinching](#) () const
- glm::vec3 [Position](#) () const
- void [AssignStructure](#) ([MovableStructure](#) \*p\_structure)
- [MovableStructure](#) \* [AssignedStructure](#) () const
- void [CleanUp](#) ()

## Private Member Functions

- Leap::Vector [AdaptToMode](#) (Leap::Vector right\_hand\_pos, bool mode)

## Private Attributes

- Leap::Controller [d\\_controller](#)  
*The object allowing us to get data from the Leapmotion device.*
- glm::vec3 [d\\_position](#)  
*The position of the [LeapmotionPointer](#).*
- [MovableStructure](#) \* [d\\_p\\_structure](#) = NULL
- glm::mat4 [d\\_invertedWorldMatrix](#) = glm::mat4()
- bool [d\\_init](#) = false  
*Set to true once initialized.*
- [Cube](#) \* [d\\_p\\_pointerModel](#)
- [Cube](#) \* [d\\_p\\_referencePointerModel](#)  
*the reference [Cube](#) that doesn't change size*
- [Quad](#) \* [d\\_p\\_shadow](#)  
*the [Quad](#) projected on the floor, helping perceive depth*
- [GUI](#) \* [d\\_p\\_gui](#)

### 5.7.1 Detailed Description

The [LeapmotionPointer](#) is an object allowing to interact with the virtual environment by using the Leapmotion, a gesture-recognition device. It uses the pinching gesture to grab [MovableStructures](#)

### 5.7.2 Member Function Documentation

#### 5.7.2.1 Leap::Vector LeapmotionPointer::AdaptToMode ( Leap::Vector *right\_hand\_pos*, bool *mode* ) [private]

Adapts the offset, the limits and the sensitivity of the pointer depending on the current mode

- `right_hand_pos` The current position of the user's right-most hand
- `mode` The current [Simulator](#) mode

#### 5.7.2.2 `MovableStructure * LeapmotionPointer::AssignedStructure ( ) const`

Returns a pointer to the currently assigned [MovableStructure](#)

#### 5.7.2.3 `void LeapmotionPointer::AssignStructure ( MovableStructure * p_structure )`

Assigns a [MovableStructure](#) to the pointer

- `p_structure` A pointer to the desired [MovableStructure](#)

#### 5.7.2.4 `void LeapmotionPointer::CleanUp ( )`

Cleans up the three models used to represent the Pointer

#### 5.7.2.5 `void LeapmotionPointer::Draw ( const glm::mat4 & VP ) const`

Draws the [LeapmotionPointer](#)'s Model

#### 5.7.2.6 `void LeapmotionPointer::Init ( GUI * p_gui )`

Initializes the [LeapmotionPointer](#) its offset, its position (set to (0,0,0)) and its Model

- `p_gui` A pointer to the [GUI](#) that uses the pointer

#### 5.7.2.7 `bool LeapmotionPointer::Pinching ( ) const`

returns whether or not the user's rightmost hand is pinching

#### 5.7.2.8 `glm::vec3 LeapmotionPointer::Position ( ) const`

Returns the position of the user's rightmost hand or the last captured position if the hand is out of reach

#### 5.7.2.9 `void LeapmotionPointer::update ( bool mode )`

Updates the the [LeapmotionPointer](#)'s position and drags the pinched [Structure](#) if there is one

- `mode` The current [Simulator](#) mode

#### 5.7.2.10 void LeapmotionPointer::UpdateWorldMatrix ( const glm::mat4 & worldMatrix )

Updates the `_invertedWorldMatrix` with the one from the main

### 5.7.3 Member Data Documentation

#### 5.7.3.1 Leap::Controller LeapmotionPointer::d\_controller [private]

The object allowing us to get data from the Leapmotion device.

#### 5.7.3.2 bool LeapmotionPointer::d\_init = false [private]

Set to true once initialized.

#### 5.7.3.3 glm::mat4 LeapmotionPointer::d\_invertedWorldMatrix = glm::mat4() [private]

This matrix is the inverse from the [Scene's](#) WorldMatrix. It allows the user to move around in the room and move Structures more easily

#### 5.7.3.4 GUI\* LeapmotionPointer::d\_p\_gui [private]

A pointer to the [GUI](#) so it can tell it to add a new [Structure](#) to a button once it's dropped

#### 5.7.3.5 Cube\* LeapmotionPointer::d\_p\_pointerModel [private]

the [Cube](#) representing the [LeapmotionPointer](#) and changing size depending on its pinching value

#### 5.7.3.6 Cube\* LeapmotionPointer::d\_p\_referencePointerModel [private]

the reference [Cube](#) that doesn't change size

#### 5.7.3.7 Quad\* LeapmotionPointer::d\_p\_shadow [private]

the [Quad](#) projected on the floor, helping perceive depth

#### 5.7.3.8 MovableStructure\* LeapmotionPointer::d\_p\_structure = NULL [private]

A pointer to the [Structure](#) being dragged (NULL if no [Structure](#) is being dragged)



## 5.7.3.9 glm::vec3 LeapmotionPointer::d\_position [private]

The position of the [LeapmotionPointer](#).

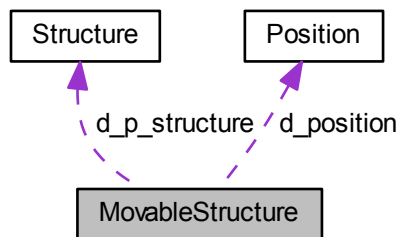
The documentation for this class was generated from the following files:

- RoombotsSimulator/[LeapmotionPointer.hh](#)
- RoombotsSimulator/[LeapmotionPointer.cc](#)

## 5.8 MovableStructure Class Reference

```
#include <MovableStructure.hh>
```

Collaboration diagram for MovableStructure:



### Public Member Functions

- [MovableStructure](#) ([Structure](#) \*p\_structure, glm::vec3 position, int ID, unsigned int buttonID)
- bool [CloseEnough](#) (glm::vec3 position) const
- bool [CloseEnough](#) (glm::vec3 position, float distance) const
- void [Drop](#) ()
- void [Drag](#) (const glm::vec3 &position)
- void [Draw](#) (const glm::mat4 &VP) const
- [Position](#) [GetPosition](#) () const
- unsigned int [LinkedButtonID](#) () const
- std::vector< [Position](#) > [RoombotsPositions](#) () const

### Private Member Functions

- void [SetCenterOffset](#) ()
- void [SetPosition](#) (glm::vec3 position)

## Private Attributes

- [Structure](#) \*const [d\\_p\\_structure](#)  
The pointer to the [Structure](#) to be moved.
- [Position](#) [d\\_position](#)  
The structure's position.
- int [d\\_ID](#)  
Its ID.
- unsigned int [d\\_buttonID](#)
- bool [d\\_moving](#) = false  
Whether the [Structure](#) is moving or not. (used in [Drop\(\)](#))

### 5.8.1 Detailed Description

This class encapsulates a Roombot [Structure](#) as something that interact with the buttons and the [Leapmotion↔ Pointer](#). Indeed, Structures are static sets of Roombots Module and are not meant to be moved around.

This encapsulation allows to use the same [Structure](#) for all similar MovableStructures. For instance, all 'chairs' MovableStructures use the same [Structure](#) through a pointer.

All the visible Structures when running the software are actually MovableStructures.

### 5.8.2 Constructor & Destructor Documentation

5.8.2.1 `MovableStructure::MovableStructure ( Structure * p_structure, glm::vec3 position, int ID, unsigned int buttonID )`

### 5.8.3 Member Function Documentation

5.8.3.1 `bool MovableStructure::CloseEnough ( glm::vec3 position ) const`

Checks if a position is close enough from the [MovableStructure's Structure's](#) center

- `position` The reference position

#### Returns

true if `position` is within the default drag radius

5.8.3.2 `bool MovableStructure::CloseEnough ( glm::vec3 position, float distance ) const`

Checks if a position is at most at a certain distance from the [MovableStructure's Structure's](#) center

- `position` The position used to check the distance

#### Returns

true if `position` is within `distance` from the center

5.8.3.3 void MovableStructure::Drag ( const glm::vec3 & *position* )

Moves the [Structure](#) to the position passed in argument

- *position* The target position where to drag the currently assigned [MovableStructure](#)

5.8.3.4 void MovableStructure::Draw ( const glm::mat4 & *VP* ) const

Draws the [Structure](#)

5.8.3.5 void MovableStructure::Drop ( )

Immediately Drops the [Structure](#) where the shadow is drawn There is no movement of the Model, it simply "teleports" on the ground

5.8.3.6 Position MovableStructure::GetPosition ( ) const

Returns the reference position of the [Structure](#)

5.8.3.7 unsigned int MovableStructure::LinkedButtonID ( ) const

Returns the ID of the button from which the [MovableStructure](#) comes

5.8.3.8 std::vector< [Position](#) > MovableStructure::RoombotsPositions ( ) const

Returns the positions of all the Roombots of its [Structure](#)

5.8.3.9 void MovableStructure::SetCenterOffset ( ) [private]

Computes the center's offset from the lower left corner

5.8.3.10 void MovableStructure::SetPosition ( glm::vec3 *position* ) [private]

Sets the new position of the the [MovableStructure](#) and ensures that it stays on the grid

## 5.8.4 Member Data Documentation

5.8.4.1 unsigned int MovableStructure::d\_buttonID [private]

The ID of the button from which the [Structure](#) was created Once dropped, it is set to -1 to "unlink" it from the button

5.8.4.2 `int MovableStructure::d_ID` `[private]`

Its ID.

5.8.4.3 `bool MovableStructure::d_moving = false` `[private]`

Whether the [Structure](#) is moving or not. (used in [Drop\(\)](#))

5.8.4.4 `Structure* const MovableStructure::d_p_structure` `[private]`

The pointer to the [Structure](#) to be moved.

5.8.4.5 `Position MovableStructure::d_position` `[private]`

The structure's position.

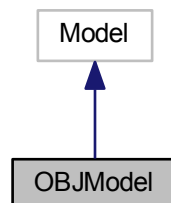
The documentation for this class was generated from the following files:

- [RoombotsSimulator/MovableStructure.hh](#)
- [RoombotsSimulator/MovableStructure.cc](#)

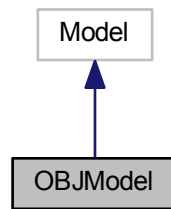
## 5.9 OBJModel Class Reference

```
#include <OBJModel.hh>
```

Inheritance diagram for OBJModel:



Collaboration diagram for OBJModel:



### Public Member Functions

- [OBJModel](#) (const std::string OBJFilename, const char \*vShaderFileName, const char \*fShaderFileName, const char \*textureFileName, const glm::vec4 &color)

### Protected Member Functions

- virtual void [SetVertices](#) (std::vector< glm::vec3 > \*vertices)
- virtual void [SetUVs](#) (std::vector< glm::vec2 > \*uvs)

### Protected Attributes

- const std::string [\\_objfilename](#)  
*The name of the .obj file to import.*

#### 5.9.1 Detailed Description

An [OBJModel](#) is a `Model` that gets its vertices from a .obj file. .obj (wavefront) files are standard graphics models files and are pretty easy to parse. But they also contain shading and texturing information that are not used in this software. Furthermore, .obj files sometimes contain polygons which would be ignored by the parser. However, quads would be interpreted as two triangles

This class allows to use complex models pretty easily.

#### 5.9.2 Constructor & Destructor Documentation

- 5.9.2.1 `OBJModel::OBJModel ( const std::string OBJFilename, const char * vShaderFileName, const char * fShaderFileName, const char * textureFileName, const glm::vec4 & color )`

Same constructor as the `Model` class but with the .obj filename added to it

- `OBJFilename` The name of the .obj file containing the [OBJModel](#)

### 5.9.3 Member Function Documentation

5.9.3.1 void OBJModel::SetUVs ( std::vector< glm::vec2 > \* *uvs* ) [protected],[virtual]

5.9.3.2 void OBJModel::SetVertices ( std::vector< glm::vec3 > \* *vertices* ) [protected],[virtual]

This method parses the .obj filename of the [OBJModel](#) and creates the corresponding vertices

### 5.9.4 Member Data Documentation

5.9.4.1 const std::string OBJModel::\_objfilename [protected]

The name of the .obj file to import.

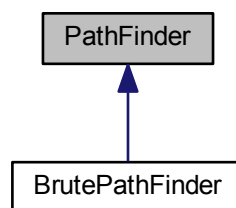
The documentation for this class was generated from the following files:

- RoombotsSimulator/[OBJModel.hh](#)
- RoombotsSimulator/[OBJModel.cc](#)

## 5.10 Pathfinder Class Reference

```
#include <PathFinder.hh>
```

Inheritance diagram for Pathfinder:



### Public Member Functions

- virtual void [Run](#) ([Path](#) &path, const [Position](#) &start, const [Position](#) &finish) const =0
- virtual std::string [Name](#) () const =0

### 5.10.1 Detailed Description

This class represents a path-finding algorithm. It is abstract, as we can design a lot of different path-finding algorithm, and all must implement the 'run' method described below.

The Path represents a series of successive Positions a Roombot Module must pass through to get to its final position

### 5.10.2 Member Function Documentation

#### 5.10.2.1 `virtual std::string PathFinder::Name ( ) const [pure virtual]`

This method simply returns the name of the path-finding algorithm

Implemented in [BrutePathFinder](#).

#### 5.10.2.2 `virtual void PathFinder::Run ( Path & path, const Position & start, const Position & finish ) const [pure virtual]`

Creates a Path, a succession of [Position](#) from a [Position](#) to another. It uses an external Path and fills it

- `path` A reference to the Path to fill
- `start` The starting [Position](#) of the Path
- `finish` The finishing [Position](#) of the Path

Implemented in [BrutePathFinder](#).

The documentation for this class was generated from the following file:

- RoombotsSimulator/[PathFinder.hh](#)

## 5.11 Position Class Reference

```
#include <Position.hh>
```

### Public Member Functions

- [Position](#) ()
- [Position](#) (int `x`, int `y`, int `z`)
- [Position](#) ([Position](#) \*)
- [Position](#) (glm::vec3)
- [Position](#) operator+ ([Position](#) other) const
- [Position](#) operator- ([Position](#) other) const
- [Position](#) operator\* (int factor) const
- void operator+= ([Position](#) other)
- void operator-= ([Position](#) other)
- void operator\*= (int factor)
- bool operator== ([Position](#) other) const
- bool operator!= ([Position](#) other) const
- int distanceTo ([Position](#) other) const
- void [Print](#) () const
- glm::vec3 [ToGLM](#) () const
- int `x` () const
- int `y` () const
- int `z` () const

## Private Attributes

- int [d\\_x](#)
- int [d\\_y](#)
- int [d\\_z](#)

### 5.11.1 Detailed Description

This class is a simple triplet of integers that represent a discrete [Position](#). It allows easier handling of grid-bound objects. The grid is also Roombot-sized. It has basic operators and methods, as expected from such object

### 5.11.2 Constructor & Destructor Documentation

#### 5.11.2.1 `Position::Position ( )`

This constructor creates a new [Position](#) at (0,0,0)

#### 5.11.2.2 `Position::Position ( int x, int y, int z )`

This constructor creates a new [Position](#) at (x,y,z)

#### 5.11.2.3 `Position::Position ( Position * other )`

Copy constructor

#### 5.11.2.4 `Position::Position ( glm::vec3 other )`

This constructor converts a 'glm::vec3' into a [Position](#) that is at the closest spot on the grid from the 'glm::vec3' passed as argument

### 5.11.3 Member Function Documentation

#### 5.11.3.1 `int Position::distanceTo ( Position other ) const`

Sums up the difference between all three coordinates of 'this' and 'other'



5.11.3.2 `bool Position::operator!= ( Position other ) const`

5.11.3.3 `Position Position::operator* ( int factor ) const`

5.11.3.4 `void Position::operator*= ( int factor )`

5.11.3.5 `Position Position::operator+ ( Position other ) const`

5.11.3.6 `void Position::operator+= ( Position other )`

5.11.3.7 `Position Position::operator- ( Position other ) const`

5.11.3.8 `void Position::operator-= ( Position other )`

5.11.3.9 `bool Position::operator== ( Position other ) const`

5.11.3.10 `void Position::Print ( ) const`

Prints out the [Position](#) as (x,y,z)

5.11.3.11 `glm::vec3 Position::ToGLM ( ) const`

Returns a 'glm::vec3' equivalent to the [Position](#)

5.11.3.12 `int Position::x ( ) const`

5.11.3.13 `int Position::y ( ) const`

5.11.3.14 `int Position::z ( ) const`

## 5.11.4 Member Data Documentation

5.11.4.1 `int Position::d_x` [private]

5.11.4.2 `int Position::d_y` [private]

5.11.4.3 `int Position::d_z` [private]

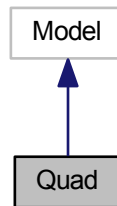
The documentation for this class was generated from the following files:

- RoombotsSimulator/[Position.hh](#)
- RoombotsSimulator/[Position.cc](#)

## 5.12 Quad Class Reference

```
#include <Quad.hh>
```

Inheritance diagram for Quad:



Collaboration diagram for Quad:



### Public Member Functions

- [Quad](#) (const std::string vShaderFileName, const std::string fShaderFileName, const std::string textureFileName, const glm::vec4 &color)
- [Quad](#) (const char \*vShaderFileName, const char \*fShaderFileName, const char \*textureFileName, const glm::vec4 &color)

### Protected Member Functions

- virtual void [SetVertices](#) (std::vector< glm::vec3 > \*vertices)
- virtual void [SetUVs](#) (std::vector< glm::vec2 > \*uvs)

### 5.12.1 Constructor & Destructor Documentation

5.12.1.1 `Quad::Quad ( const std::string vShaderFileName, const std::string fShaderFileName, const std::string textureFileName, const glm::vec4 & color ) [inline]`

5.12.1.2 `Quad::Quad ( const char * vShaderFileName, const char * fShaderFileName, const char * textureFileName, const glm::vec4 & color ) [inline]`

### 5.12.2 Member Function Documentation

5.12.2.1 `void Quad::SetUVs ( std::vector< glm::vec2 > * uvs ) [protected],[virtual]`

5.12.2.2 `void Quad::SetVertices ( std::vector< glm::vec3 > * vertices ) [protected],[virtual]`

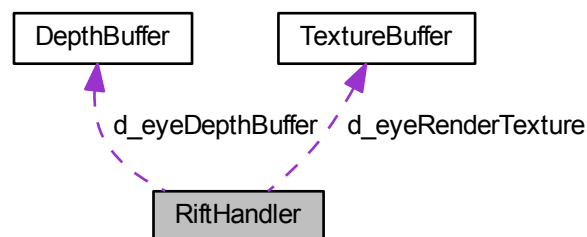
The documentation for this class was generated from the following files:

- [RoombotsSimulator/Quad.hh](#)
- [RoombotsSimulator/Quad.cc](#)

## 5.13 RiftHandler Class Reference

```
#include <RiftHandler.hh>
```

Collaboration diagram for RiftHandler:



### Public Member Functions

- void [DisplayOnRift](#) ()
- void [Init](#) ([DisplayFunction](#))
- OVR::Matrix4f [ovrViewProjMatrix](#) ()
- glm::mat4 [glmViewProjMatrix](#) ()
- unsigned int [ResolutionWidth](#) ()
- unsigned int [ResolutionHeight](#) ()
- void [CleanUp](#) ()

## Private Attributes

- OVR::Matrix4f [d\\_viewProjMatrix](#)  
*THE viewProj matrix representing the orientation of the HMD.*
- ovrHmd [d\\_hmd](#) = nullptr  
*The code object representing the Rift's HMD.*
- ovrTrackingState [d\\_trackingState](#)  
*The tracking state of the HMD.*
- bool [d\\_isVisible](#) = false  
*Keeps the rendering to be displayed while the Rift hasn't been initialized.*
- ovrEyeRenderDesc [d\\_EyeRenderDesc](#) [2]  
*Store all the rendering information for both eyes.*
- [TextureBuffer](#) \* [d\\_eyeRenderTexture](#) [2]  
*The textures where the image from both eyes will be stored.*
- [DepthBuffer](#) \* [d\\_eyeDepthBuffer](#) [2]  
*The buffer where the depth information from both eyes will be stored.*
- GLuint [d\\_mirrorFBO](#) = 0  
*The FBO for the mirror display.*
- ovrGLTexture \* [d\\_mirrorTexture](#)  
*The texture for the mirror display.*
- [DisplayFunction](#) [d\\_displayFunction](#)  
*the function that draws what will be displayed on the Rift's screen*

## 5.13.1 Member Function Documentation

### 5.13.1.1 void RiftHandler::CleanUp ( )

Destroys and shuts the Rift virtual object down

### 5.13.1.2 void RiftHandler::DisplayOnRift ( )

displays the rendering done in 'displayFunction' on the Rift's screen

### 5.13.1.3 glm::mat4 RiftHandler::glmViewProjMatrix ( )

Same as 'ovrProjViewMatrix()' but returns a glm::mat4 instead

### 5.13.1.4 void RiftHandler::Init ( [DisplayFunction](#) *disFunc* )

Initializes the Rift

- [DisplayFunction](#) the function that will do the actual rendering and will draw what is to be displayed on the Rift's screen

#### 5.13.1.5 `OVR::Matrix4f RiftHandler::ovrViewProjMatrix ( )`

Returns the view projection matrix that is built based on the Rift's orientation

#### 5.13.1.6 `unsigned int RiftHandler::ResolutionHeight ( )`

Returns the height of the Rift's resolution

#### 5.13.1.7 `unsigned int RiftHandler::ResolutionWidth ( )`

Returns the width of the Rift's resolution

### 5.13.2 Member Data Documentation

#### 5.13.2.1 `DisplayFunction RiftHandler::d_displayFunction` `[private]`

the function that draws what will be displayed on the Rift's screen

#### 5.13.2.2 `DepthBuffer* RiftHandler::d_eyeDepthBuffer[2]` `[private]`

The buffer where the depth information from both eyes will be stored.

#### 5.13.2.3 `ovrEyeRenderDesc RiftHandler::d_EyeRenderDesc[2]` `[private]`

Store all the rendering information for both eyes.

#### 5.13.2.4 `TextureBuffer* RiftHandler::d_eyeRenderTexture[2]` `[private]`

The textures where the image from both eyes will be stored.

#### 5.13.2.5 `ovrHmd RiftHandler::d_hmd = nullptr` `[private]`

The code object representing the Rift's HMD.

#### 5.13.2.6 `bool RiftHandler::d_isVisible = false` `[private]`

Keeps the rendering to be displayed while the Rift hasn't been initialized.

#### 5.13.2.7 `GLuint RiftHandler::d_mirrorFBO = 0` `[private]`

The FBO for the mirror display.

#### 5.13.2.8 `ovrGLTexture* RiftHandler::d_mirrorTexture` [private]

The texture for the mirror display.

#### 5.13.2.9 `ovrTrackingState RiftHandler::d_trackingState` [private]

The tracking state of the HMD.

#### 5.13.2.10 `OVR::Matrix4f RiftHandler::d_viewProjMatrix` [private]

THE viewProj matrix representing the orientation of the HMD.

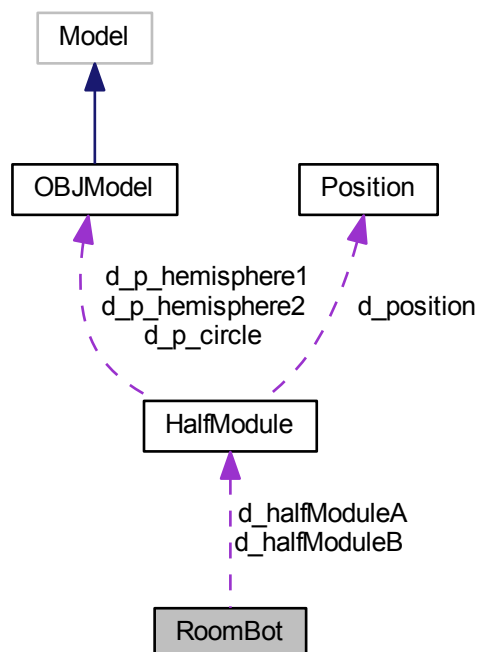
The documentation for this class was generated from the following files:

- RoombotsSimulator/[RiftHandler.hh](#)
- RoombotsSimulator/[RiftHandler.cc](#)

## 5.14 RoomBot Class Reference

```
#include <RoomBot.hh>
```

Collaboration diagram for RoomBot:



## Public Member Functions

- [RoomBot](#) ([Position](#) A, [Position](#) B, [OBJModel](#) \*p\_h1, [OBJModel](#) \*p\_h2, [OBJModel](#) \*p\_circle)
- [RoomBot](#) (int Ax, int Ay, int Az, int Bx, int By, int Bz, [OBJModel](#) \*p\_h1, [OBJModel](#) \*p\_h2, [OBJModel](#) \*p\_circle)
- glm::vec3 [MiddlePosition](#) () const
- void [Draw](#) (const glm::mat4 &VP) const
- [Position](#) [PositionA](#) () const
- [Position](#) [PositionB](#) () const

## Private Attributes

- [HalfModule](#) d\_halfModuleA  
*The first half of the Roombot.*
- [HalfModule](#) d\_halfModuleB  
*The second half of the Roombot.*

### 5.14.1 Detailed Description

The Roombot class is a simple class that encapsulates two halfModules. It is mainly an interface class between the HalfModules and other parts of the software

### 5.14.2 Constructor & Destructor Documentation

#### 5.14.2.1 [RoomBot::RoomBot](#) ( [Position](#) A, [Position](#) B, [OBJModel](#) \* p\_h1, [OBJModel](#) \* p\_h2, [OBJModel](#) \* p\_circle )

Creates a new [RoomBot](#)

- A The [Position](#) of the first [HalfModule](#)
- B The [Position](#) of the second [HalfModule](#)
- p\_h1,p\_h2\_p\_circle Pointers to [OBJModel](#) used to construct the two [HalfModule](#)

#### 5.14.2.2 [RoomBot::RoomBot](#) ( int Ax, int Ay, int Az, int Bx, int By, int Bz, [OBJModel](#) \* p\_h1, [OBJModel](#) \* p\_h2, [OBJModel](#) \* p\_circle )

Same as the first constructor but with the six `int` used to create the two necessary [Position](#)

### 5.14.3 Member Function Documentation

#### 5.14.3.1 void [RoomBot::Draw](#) ( const glm::mat4 & VP ) const

Draws the [RoomBot](#)

#### 5.14.3.2 `glm::vec3 RoomBot::MiddlePosition ( ) const` `[inline]`

Returns the position of the middle between the two HalfModules

#### 5.14.3.3 `Position RoomBot::PositionA ( ) const`

Returns the [Position](#) of the first [HalfModule](#)

#### 5.14.3.4 `Position RoomBot::PositionB ( ) const`

Returns the [Position](#) of the second [HalfModule](#)

### 5.14.4 Member Data Documentation

#### 5.14.4.1 `HalfModule RoomBot::d_halfModuleA` `[private]`

The first half of the Roombot.

#### 5.14.4.2 `HalfModule RoomBot::d_halfModuleB` `[private]`

The second half of the Roombot.

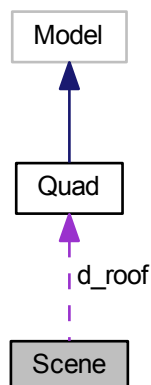
The documentation for this class was generated from the following files:

- RoombotsSimulator/[RoomBot.hh](#)
- RoombotsSimulator/[RoomBot.cc](#)

## 5.15 Scene Class Reference

```
#include <Scene.hh>
```

Collaboration diagram for Scene:





## Public Member Functions

- void [Init](#) (float roomSize)
- void const [Render](#) (const glm::mat4 &VP, bool drawRoof)
- void [CleanUp](#) ()

## Private Member Functions

- void [AddModel](#) (Model \*sourceModel)

## Private Attributes

- std::vector< const Model \* > [d\\_models](#)  
*The models contained in the scene.*
- size\_t [d\\_nModels](#)  
*The number of models in the scene.*
- Quad \* [d\\_roof](#)  
*the roof*

### 5.15.1 Detailed Description

The [Scene](#) contains all the Models used to represent the scene in which the simulation takes place. The floor, the walls, the windows, the roof, the door and a skybox surrounding the room. Aside from the roof, all Models are added polymorphically. The roof gets a special treatment because it is not necessarily drawn, depending on the current mode.

### 5.15.2 Member Function Documentation

#### 5.15.2.1 void Scene::AddModel ( Model \* *sourceModel* ) [private]

Adds a Model to the [Scene](#)

- *sourceModel* The Model to add

#### 5.15.2.2 void Scene::CleanUp ( )

Cleans up all the Models of the [Scene](#)

#### 5.15.2.3 void Scene::Init ( float *roomSize* )

Initializes the [Scene](#)

- *roomSize* the size of the room in the middle of the [Scene](#)

#### 5.15.2.4 void const Scene::Render ( const glm::mat4 & VP, bool drawRoof )

Drawns all the elements of the [Scene](#) and the roof if the current mode is RoomView

- VP the Projection-View matrix
- drawRoof whether or not the roof should be drawn

### 5.15.3 Member Data Documentation

#### 5.15.3.1 std::vector<const Model\*> Scene::d\_models [private]

The models contained in the scene.

#### 5.15.3.2 size\_t Scene::d\_nModels [private]

The number of models in the scene.

#### 5.15.3.3 Quad\* Scene::d\_roof [private]

the roof

The documentation for this class was generated from the following files:

- RoombotsSimulator/[Scene.hh](#)
- RoombotsSimulator/[Scene.cc](#)

## 5.16 ShaderLoader Class Reference

```
#include <ShaderLoader.hh>
```

### Public Member Functions

- GLuint [CreateProgram](#) (const char \*VertexShaderFilename, const char \*FragmentShaderFilename)

### Static Public Member Functions

- static std::string [DefaultVertexShader](#) ()
- static std::string [DefaultFragmentShader](#) ()

### Private Member Functions

- std::string [ReadShader](#) (const char \*filename)
- GLuint [CreateShader](#) (GLenum shaderType, std::string source, char \*shaderName)

### 5.16.1 Member Function Documentation

#### 5.16.1.1 GLuint ShaderLoader::CreateProgram ( const char \* *VertexShaderFilename*, const char \* *FragmentShaderFilename* )

creates a new program using the two shaders indicated by the names passed in arguments and returns its ID

##### Returns

the GLuint representing the new program

#### 5.16.1.2 GLuint ShaderLoader::CreateShader ( GLenum *shaderType*, std::string *source*, char \* *shaderName* ) [private]

creates a new shader

- *shaderType* the type of shader to create
- *source* the string containing the shader program
- *shaderName* the name of the shader

##### Returns

the GLuint representing the new shader

#### 5.16.1.3 std::string ShaderLoader::DefaultFragmentShader ( ) [static]

Returns the following simple fragment shader :

```
#version 330 core
uniform sampler2D tex;
in vec2 uv;
out vec4 color;
void main(){
color = vec4(1.0,1.0,1.0,1.0);
}
```

It draws the whole model in plain opaque white

#### 5.16.1.4 std::string ShaderLoader::DefaultVertexShader ( ) [static]

Returns the following simple vertex shader :

```
#version 330 core
uniform mat4 MVP;
in vec3 vpoint;
in vec2 vtxcoord;
out vec2 uv;
void main(){
gl_Position = MVP * vec4(vpoint,1.0);
uv = vtxcoord;
}
```

It simply applies the MVP matrix to the vertex' position

#### 5.16.1.5 `std::string ShaderLoader::ReadShader ( const char * filename )` `[private]`

reads a glsl program from a file

- `filename` the name of the file containing the glsl program

##### Returns

a `string` containing the whole file or "invalidShader" if the file couldn't be read

The documentation for this class was generated from the following files:

- RoombotsSimulator/[ShaderLoader.hh](#)
- RoombotsSimulator/[ShaderLoader.cc](#)

## 5.17 Simulation Class Reference

```
#include <Simulation.hh>
```

### Public Member Functions

- void [Initialize](#) (const std::vector< [Position](#) > roombotsFinalPositions, [PathFinder](#) \*pathFinder)
- bool [NextStep](#) ()
- void [Draw](#) (const glm::mat4 &VP)
- bool [IsOver](#) ()
- bool [IsInitialized](#) ()
- void [Run](#) ()

### Private Member Functions

- void [Reset](#) ()

### Private Attributes

- std::vector< [Path](#) > [d\\_paths](#)  
*The vector of all the Roombots' path.*
- std::vector< [HalfModule](#) > [d\\_halfModules](#)  
*The vector of all halfModules that will move during [Simulation](#).*
- unsigned int [d\\_currentStep](#) = 0  
*The index of the current simulation step.*
- bool [d\\_init](#) = false  
*Whether or not the [Simulation](#) has been initialized.*
- bool [d\\_over](#) = true  
*Whether or not the [Simulation](#) is over. It is considered over when not initialized.*
- std::clock\_t [d\\_refClock](#)  
*The reference clock used to time the calls to 'nextStep()'.*

### 5.17.1 Detailed Description

The [Simulation](#) encapsulates all that is needed to simulate the movements of the Roombot modules and allows to see how they would move around in the room to take their final place in the scene the User would have previously set up. It runs in parallel with the rest of the software once initialized but does not use any concurrent feature such as "std::thread".

IMPORTANT NOTE : All modules and their path must be retrieve outside of the [Simulation](#) and must be given to it when initialized.

### 5.17.2 Member Function Documentation

#### 5.17.2.1 void Simulation::Draw ( const glm::mat4 & VP )

Draws all modules needed to perform the [Simulation](#)

#### 5.17.2.2 void Simulation::Initialize ( const std::vector< [Position](#) > roombotsFinalPositions, [PathFinder](#) \* pathFinder )

Initializes the [Simulation](#).

- roombotsFinalPositions A vector of all the Positions of all the Roombot
- pathFinder The [PathFinder](#) to use to compute all the Path

#### 5.17.2.3 bool Simulation::IsInitialized ( ) [inline]

Returns whether or not the [Simulation](#) has been initialized

#### 5.17.2.4 bool Simulation::IsOver ( ) [inline]

Returns whether or not the simulation is over.

#### 5.17.2.5 bool Simulation::NextStep ( )

Executes a step of the [Simulation](#). Returns false if the simulation is over and true otherwise

#### 5.17.2.6 void Simulation::Reset ( ) [private]

Resets the [Simulation](#)

#### 5.17.2.7 void Simulation::Run ( )

Executes one steps of the [Simulation](#) if it's not over and ensures the execution of the successive steps is well-timed

### 5.17.3 Member Data Documentation

#### 5.17.3.1 `unsigned int Simulation::d_currentStep = 0` [private]

The index of the current simulation step.

#### 5.17.3.2 `std::vector<HalfModule> Simulation::d_halfModules` [private]

The vector of all halfModules that will move during [Simulation](#).

#### 5.17.3.3 `bool Simulation::d_init = false` [private]

Whether or not the [Simulation](#) has been initialized.

#### 5.17.3.4 `bool Simulation::d_over = true` [private]

Whether or not the [Simulation](#) is over. It is considered over when not initialized.

#### 5.17.3.5 `std::vector<Path> Simulation::d_paths` [private]

The vector of all the Roombots' path.

#### 5.17.3.6 `std::clock_t Simulation::d_refClock` [private]

The reference clock used to time the calls to 'nextStep()'.

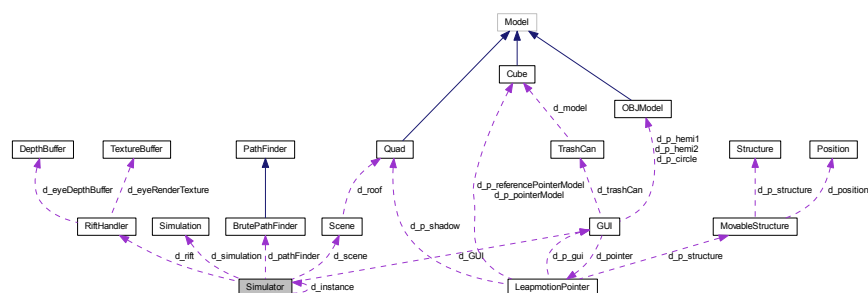
The documentation for this class was generated from the following files:

- RoombotsSimulator/[Simulation.hh](#)
- RoombotsSimulator/[Simulation.cc](#)

## 5.18 Simulator Class Reference

```
#include <Simulator.hh>
```

Collaboration diagram for Simulator:



## Public Member Functions

- void [Init](#) (int argc, char \*\*argv, [DisplayFunction](#) display, [DisplayFunction](#) renderScene, void(\*keyboardFunc)(unsigned char, int, int), void(\*resizeFunc)(int, int), void(\*closeFunc)())
- void [Start](#) ()
- void [CleanUp](#) ()
- void [Resize](#) (int w, int h)
- void [RenderScene](#) ()
- void [Display](#) ()
- void [HandleKeyboard](#) (unsigned char key, int x, int y)
- void [Close](#) ()
- glm::mat4 [WorldViewMatrix](#) ()

## Static Public Member Functions

- static [Simulator](#) & [Instance](#) ()

## Private Member Functions

- [Simulator](#) ()
- [~Simulator](#) ()
- void [Forward](#) ()
- void [Left](#) ()
- void [Backwards](#) ()
- void [Right](#) ()
- void [SwitchViewMode](#) ()
- void [InitRift](#) ([DisplayFunction](#) function)
- void [InitSimulation](#) ()
- void [MainLoop](#) ()

## Private Attributes

- [BrutePathFinder](#) [d\\_pathFinder](#)  
*The pathFinder used to create all the paths before [Simulation](#).*
- [Simulation](#) [d\\_simulation](#)  
*The [Simulation](#) that will run once the scene is set up.*
- [Scene](#) [d\\_scene](#)  
*The scene containing all the static elements of the environment.*
- [RiftHandler](#) [d\\_rift](#)  
*The object allowing easy use of the Oculus Rift.*
- [GUI](#) [d\\_GUI](#)  
*The Graphic User Interface allowing interaction with the environment.*
- unsigned int [d\\_width](#) = 0  
*The window's width.*
- unsigned int [d\\_height](#) = 0  
*The window's height.*
- glm::mat4 [d\\_worldMatrix](#) = glm::mat4()  
*The worldMatrix that changes depending on the current mode.*
- bool [d\\_mode](#) = true  
*viewing mode. false for "in-room" view, true for "box" view*
- bool [d\\_running](#) = true  
*Whether or not the [Simulator](#) is running.*
- int [d\\_windowID](#)  
*The OpenGL context window's ID.*

## Static Private Attributes

- static `Simulator d_instance = Simulator()`  
*The static instance of the `Simulator`, making it a singleton.*

### 5.18.1 Detailed Description

The `Simulator` class is the main class of the software and binds everything together. It is a singleton, as there's no need for multiple instances of it. It mainly contains the `Scene` and the Graphic User Interface and is in charge of handling all windowing system-related components, as well as handling the Oculus Rift and initializing the `Simulation` and run it when ready.

About the mode : The environment can interacted with through two modes, the 'Room-View' mode and the 'Box-View' mode. The first one is just as if the User was sitting in the room and the second is just as if the User was sitting in front of a small box containing the scene. The Room-View mode allows to move freely in the room by using the "WASD" keys and the Box-View mode allows to grab and drop Structures from containers called "Buttons"

### 5.18.2 Constructor & Destructor Documentation

#### 5.18.2.1 `Simulator::Simulator ( )` [private]

The constructor is private to ensure the singleton properties

#### 5.18.2.2 `Simulator::~~Simulator ( )` [private]

### 5.18.3 Member Function Documentation

#### 5.18.3.1 `void Simulator::Backwards ( )` [private]

#### 5.18.3.2 `void Simulator::CleanUp ( )`

Cleans up everything

#### 5.18.3.3 `void Simulator::Close ( )`

#### 5.18.3.4 `void Simulator::Display ( )`

Displays the rendered scene into the Oculus Rift

#### 5.18.3.5 `void Simulator::Forward ( )` [private]

#### 5.18.3.6 `void Simulator::HandleKeyboard ( unsigned char key, int x, int y )`

Handles the keystrokes. IMPORTANT NOTE : This is based on the value of the pressed key on a QWERTZ keyboard.



**5.18.3.7** `void Simulator::Init ( int argc, char ** argv, DisplayFunction display, DisplayFunction renderScene, void(*)(unsigned char, int, int) keyboardFunc, void(*)(int, int) resizeFunc, void(*)() closeFunc )`

Initializes the [Simulator](#) by passing the various callback functions as argument.

- `argc,argv` The first two are passed to the OpenGL context creation function.
- `display` The method that will be called at every rendering loop of the OpenGL context.
- `renderScene` The method that will be called everytime the scene has to be drawn
- `keyboardFunc` The method that handles keystrokes
- `resizeFunc` The method called everytime the window is resized

**5.18.3.8** `void Simulator::InitRift ( DisplayFunction function )` [private]

Initializes the [Scene](#) Initializes the Oculus Rift

**5.18.3.9** `void Simulator::InitSimulation ( )` [private]

Initializes the [Simulation](#)

**5.18.3.10** `Simulator & Simulator::Instance ( )` [static]

Returns the unique instance of the singleton [Simulator](#)

**5.18.3.11** `void Simulator::Left ( )` [private]

**5.18.3.12** `void Simulator::MainLoop ( )` [private]

this method allows us to have control over the main OpenGL context loop. we call one iteration of the loop ourself

**5.18.3.13** `void Simulator::RenderScene ( )`

Renders everything

**5.18.3.14** `void Simulator::Resize ( int w, int h )`

Gets called when the windows is resized. It forces the window to a certain size

**5.18.3.15** `void Simulator::Right ( )` [private]

**5.18.3.16** `void Simulator::Start ( )`

Starts the [Simulator](#)

5.18.3.17 `void Simulator::SwitchViewMode ( ) [private]`

Switches between modes

5.18.3.18 `glm::mat4 Simulator::WorldViewMatrix ( )`

Returns the world matrix

## 5.18.4 Member Data Documentation

5.18.4.1 `GUI Simulator::d_GUI [private]`

The Graphic User Interface allowing interaction with the environment.

5.18.4.2 `unsigned int Simulator::d_height = 0 [private]`

The window's height.

5.18.4.3 `Simulator Simulator::d_instance = Simulator() [static], [private]`

The static instance of the [Simulator](#), making it a singleton.

5.18.4.4 `bool Simulator::d_mode = true [private]`

viewing mode. false for "in-room" view, true for "box" view

5.18.4.5 `BrutePathFinder Simulator::d_pathFinder [private]`

The pathFinder used to create all the paths before [Simulation](#).

5.18.4.6 `RiftHandler Simulator::d_rift [private]`

The object allowing easy use of the Oculus Rift.

5.18.4.7 `bool Simulator::d_running = true [private]`

Whether or not the [Simulator](#) is running.

5.18.4.8 `Scene Simulator::d_scene [private]`

The scene containing all the static elements of the environment.

#### 5.18.4.9 Simulation Simulator::d\_simulation [private]

The [Simulation](#) that will run once the scene is set up.

#### 5.18.4.10 unsigned int Simulator::d\_width = 0 [private]

The window's width.

#### 5.18.4.11 int Simulator::d\_windowID [private]

The OpenGL context window's ID.

#### 5.18.4.12 glm::mat4 Simulator::d\_worldMatrix = glm::mat4() [private]

The worldMatrix that changes depending on the current mode.

The documentation for this class was generated from the following files:

- RoombotsSimulator/[Simulator.hh](#)
- RoombotsSimulator/[Simulator.cc](#)

## 5.19 Structure Class Reference

```
#include <Structure.hh>
```

### Public Member Functions

- [Structure](#) (std::string sourceFilename, [OBJModel](#) \*p\_h1, [OBJModel](#) \*p\_h2, [OBJModel](#) \*p\_circle)
- void [Draw](#) (const glm::mat4 &VP) const
- glm::vec3 [CenterOffset](#) () const
- std::vector< [Position](#) > [RoombotsPositions](#) () const

### Private Member Functions

- void [SetCenterOffset](#) ()

### Private Attributes

- const std::string [d\\_filename](#) = ""  
*The .rbs file name from which the [Structure](#) is loaded.*
- glm::vec3 [d\\_centerOffset](#)  
*The difference between the [Structure](#)'s position and its center.*
- std::vector< [RoomBot](#) > [d\\_roomBots](#)  
*The [RoomBot](#) modules of the [Structure](#).*

### 5.19.1 Detailed Description

This class represents a set of Roombot Modules organized to form a particular structure. It is created from .rbs files, files that contain a list of positions for the Roombot modules. They take the form :

```
0 0 0
0 1 0
0 0 1
0 1 1
...
```

Where every pair of triplet is interpreted as a Roombot module.

All coordinates are relative to the [Structure](#) and the position (0,0,0) corresponds to the left-most, lowest, closest Roombot module.

IMPORTANT NOTE : There is no verification of the validity of the .rbs file. A wrongly-written file would create a physically impossible [Structure](#).

### 5.19.2 Constructor & Destructor Documentation

5.19.2.1 `Structure::Structure ( std::string sourceFilename, OBJModel * p_h1, OBJModel * p_h2, OBJModel * p_circle )`

Imports a new [OBJModel](#) from a .obj file

- `sourceFilename` The name of the .bj file containing the model to import
- `p_h1,p_h2,p_circle` Pointers to the OBJModels needed by the [RoomBot](#) of the [Structure](#)

### 5.19.3 Member Function Documentation

5.19.3.1 `glm::vec3 Structure::CenterOffset ( ) const`

Returns the [Structure](#)'s center position which is an average of the roombots' positions

5.19.3.2 `void Structure::Draw ( const glm::mat4 & VP ) const`

Draws the [Structure](#)

5.19.3.3 `std::vector< Position > Structure::RoombotsPositions ( ) const`

Returns a vector of all the RoomBots' half-module's positions as AB AB AB...

5.19.3.4 `void Structure::SetCenterOffset ( ) [private]`

Computes the offset between the lower, closer left corner and the center of the [Structure](#). The center is calculated as the average of all of the Roombots' positions

### 5.19.4 Member Data Documentation

#### 5.19.4.1 glm::vec3 Structure::d\_centerOffset [private]

The difference between the [Structure](#)'s position and its center.

#### 5.19.4.2 const std::string Structure::d\_filename = "" [private]

The .rbs file name from which the [Structure](#) is loaded.

#### 5.19.4.3 std::vector<RoomBot> Structure::d\_roomBots [private]

The [RoomBot](#) modules of the [Structure](#).

The documentation for this class was generated from the following files:

- RoombotsSimulator/[Structure.hh](#)
- RoombotsSimulator/[Structure.cc](#)

## 5.20 TextureBuffer Struct Reference

```
#include <TextureBuffer.hh>
```

### Public Member Functions

- [TextureBuffer](#) (ovrHmd hmd, bool rendertarget, bool displayableOnHmd, OVR::Sizei size, int mipLevels, unsigned char \*data, int sampleCount)
- OVR::Sizei [GetSize](#) (void) const
- void [SetAndClearRenderSurface](#) ([DepthBuffer](#) \*dbuffer)
- void [UnsetRenderSurface](#) ()
- [TextureBuffer](#) (ovrHmd hmd, bool rendertarget, bool displayableOnHmd, OVR::Sizei size, int mipLevels, unsigned char \*data, int sampleCount)
- OVR::Sizei [GetSize](#) (void) const
- void [SetAndClearRenderSurface](#) ([DepthBuffer](#) \*dbuffer)
- void [UnsetRenderSurface](#) ()

### Public Attributes

- ovrSwapTextureSet \* [TextureSet](#)
- GLuint [texId](#)
- GLuint [fbId](#)
- OVR::Sizei [texSize](#)

## 5.20.1 Constructor & Destructor Documentation

5.20.1.1 `TextureBuffer::TextureBuffer ( ovrHmd hmd, bool rendertarget, bool displayableOnHmd, OVR::Sizei size, int mipLevels, unsigned char * data, int sampleCount )` `[inline]`

5.20.1.2 `TextureBuffer::TextureBuffer ( ovrHmd hmd, bool rendertarget, bool displayableOnHmd, OVR::Sizei size, int mipLevels, unsigned char * data, int sampleCount )` `[inline]`

## 5.20.2 Member Function Documentation

5.20.2.1 `OVR::Sizei TextureBuffer::GetSize ( void ) const` `[inline]`

5.20.2.2 `OVR::Sizei TextureBuffer::GetSize ( void ) const` `[inline]`

5.20.2.3 `void TextureBuffer::SetAndClearRenderSurface ( DepthBuffer * dbuffer )` `[inline]`

5.20.2.4 `void TextureBuffer::SetAndClearRenderSurface ( DepthBuffer * dbuffer )` `[inline]`

5.20.2.5 `void TextureBuffer::UnsetRenderSurface ( )` `[inline]`

5.20.2.6 `void TextureBuffer::UnsetRenderSurface ( )` `[inline]`

## 5.20.3 Member Data Documentation

5.20.3.1 `GLuint TextureBuffer::fbold`

5.20.3.2 `GLuint TextureBuffer::texId`

5.20.3.3 `OVR::Sizei TextureBuffer::texSize`

5.20.3.4 `ovrSwapTextureSet * TextureBuffer::TextureSet`

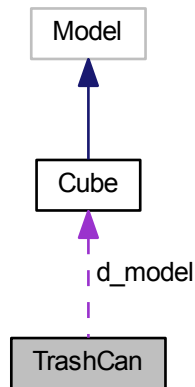
The documentation for this struct was generated from the following files:

- RoombotsSimulator/[TextureBuffer.cc](#)
- RoombotsSimulator/[TextureBuffer.hh](#)

## 5.21 TrashCan Class Reference

```
#include <TrashCan.hh>
```

Collaboration diagram for TrashCan:



### Public Member Functions

- [TrashCan](#) (glm::vec3 position)
- void [Draw](#) (const glm::mat4 &VP) const
- void [CleanUp](#) () const
- glm::vec3 [Position](#) () const

### Private Attributes

- [Cube](#) \* [d\\_model](#)  
*the cube model used to represent the trashcan*
- const glm::vec3 [d\\_position](#)  
*its position within the scene*

#### 5.21.1 Detailed Description

The [TrashCan](#) gives a way to remove MovableStructures from the [Scene](#) by dropping inside of it.

#### 5.21.2 Constructor & Destructor Documentation

##### 5.21.2.1 TrashCan::TrashCan ( glm::vec3 position )

Creates a new [TrashCan](#)

- `position` The position of the new [TrashCan](#)

### 5.21.3 Member Function Documentation

#### 5.21.3.1 void TrashCan::CleanUp ( ) const

Cleans up the Model representing the [TrashCan](#)

#### 5.21.3.2 void TrashCan::Draw ( const glm::mat4 & VP ) const

Draws the [TrashCan](#)

#### 5.21.3.3 glm::vec3 TrashCan::Position ( ) const [inline]

Returns the trashCan's position

### 5.21.4 Member Data Documentation

#### 5.21.4.1 Cube\* TrashCan::d\_model [private]

the cube model used to represent the trashcan

#### 5.21.4.2 const glm::vec3 TrashCan::d\_position [private]

its position within the scene

The documentation for this class was generated from the following files:

- RoombotsSimulator/[TrashCan.hh](#)
- RoombotsSimulator/[TrashCan.cc](#)



## Chapter 6

# File Documentation

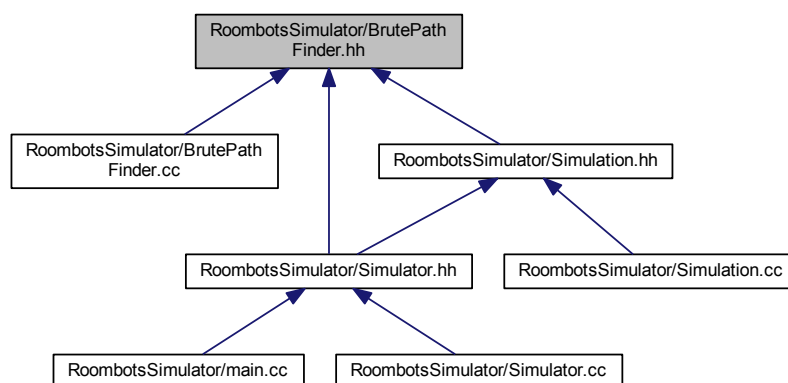
### 6.1 RoombotsSimulator/BrutePathFinder.cc File Reference

```
#include "BrutePathFinder.hh"
```

### 6.2 RoombotsSimulator/BrutePathFinder.hh File Reference

```
#include "PathFinder.hh"
```

This graph shows which files directly or indirectly include this file:



### Classes

- class [BrutePathFinder](#)

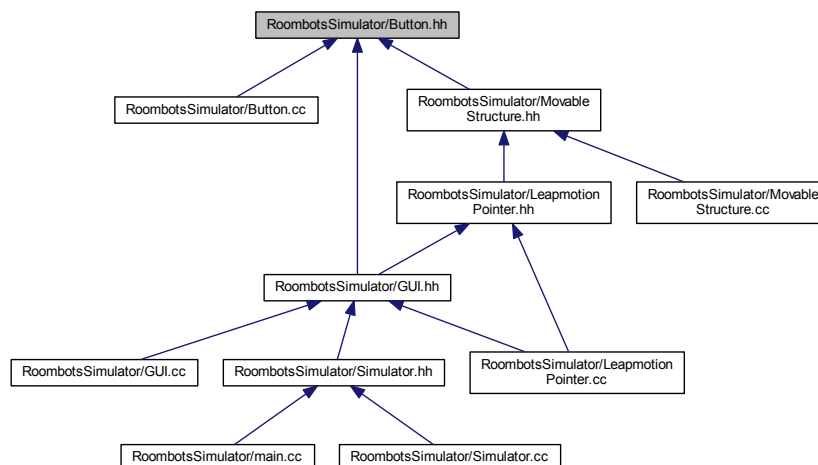
### 6.3 RoombotsSimulator/Button.cc File Reference

```
#include "Button.hh"
#include "Structure.hh"
```

### 6.4 RoombotsSimulator/Button.hh File Reference

```
#include "Cube.hh"
#include "Quad.hh"
#include "Structure.hh"
```

This graph shows which files directly or indirectly include this file:



#### Classes

- class [Button](#)

### 6.5 RoombotsSimulator/common.hh File Reference

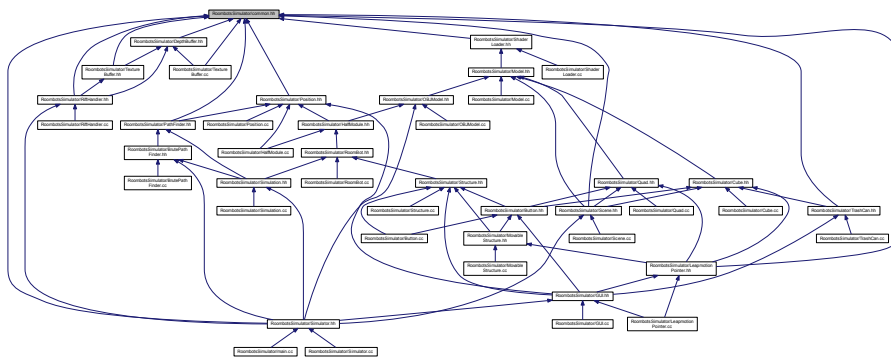
```
#include "Libraries\glew\glew.h"
```

```

#include "Libraries\glew\glew.h"
#include "Libraries\freeglut\freeglut.h"
#include "glm\glm.hpp"
#include "glm\gtc\matrix_transform.hpp"
#include "glm\gtx\transform.hpp"
#include "glm\gtx\euler_angles.hpp"
#include "Libraries\soil\SOIL.h"
#include "Libraries\wgl\wglext.h"
#include "Libraries\wgl\glext.h"
#include "Libraries\OVR\OVR_CAPI_GL.h"
#include "Libraries\OVR\Extras\OVR_Math.h"
#include <Windows.h>
#include <iostream>
#include <iomanip>
#include <string>
#include <fstream>
#include <vector>
#include <cmath>
#include <ctime>
#include "Libraries\Leap\Leap.h"

```

This graph shows which files directly or indirectly include this file:



## Macros

- `#define EYES_POSITION 1.2f`  
*This file simply regroups includes of external libraries used in most other files to leave said files clean.*
- `#define MODULE_SIZE 0.12f`  
*The size of a a half of a Roombots module.*
- `#define ROOM_SIZE 5.0f`  
*The size of the room.*
- `#define BUTTON_SIZE 0.5f`  
*The size of a button.*
- `#define BUTTON_SEPARATION 1.0f`  
*The horizontal distance between two buttons.*
- `#define BUTTON_UP_START 1.8f`  
*The vertical point where the buttons are positioned.*
- `#define BUTTON_DEPTH_OFFSET -4.5f`  
*The depth point where the buttons are positioned.*
- `#define BUTTON_RIGHT_START 2.0f`  
*The horizontal point from where the buttons are drawn.*

- `#define LEAP_POINTER_SIZE 1.0f`  
The size of the *LeapmotionPointer*.
- `#define COORDINATE_SYSTEM_SCALE_CONVERSION 0.005f`
- `#define BOX_COORDINATE_SYSTEM_SCALE_CONVERSION 0.016f`
- `#define PINCHING_LIMIT 0.7f`  
The minimal pinching value to consider that the hand is pinching.
- `#define DRAG_RADIUS 0.4f`  
The minimal distance to grab a *Structure*.
- `#define TRASH_CAN_SIZE 2.0f`  
The size of the *TrashCan*.

## 6.5.1 Macro Definition Documentation

### 6.5.1.1 `#define BOX_COORDINATE_SYSTEM_SCALE_CONVERSION 0.016f`

A conversion factor to scale the data coming from the Leapmotion device to fit the Box-View mode characteristics

### 6.5.1.2 `#define BUTTON_DEPTH_OFFSET -4.5f`

The depth point where the buttons are positioned.

### 6.5.1.3 `#define BUTTON_RIGHT_START 2.0f`

The horizontal point from where the buttons are drawn.

### 6.5.1.4 `#define BUTTON_SEPARATION 1.0f`

The horizontal distance between two buttons.

### 6.5.1.5 `#define BUTTON_SIZE 0.5f`

The size of a button.

### 6.5.1.6 `#define BUTTON_UP_START 1.8f`

The vertical point where the buttons are positioned.

### 6.5.1.7 `#define COORDINATE_SYSTEM_SCALE_CONVERSION 0.005f`

A conversion factor to scale the data coming from the Leapmotion device to fit the Room-View mode characteristics

#### 6.5.1.8 `#define DRAG_RADIUS 0.4f`

The minimal distance to grab a [Structure](#).

#### 6.5.1.9 `#define EYES_POSITION 1.2f`

This file simply regroups includes of external libraries used in most other files to leave said files clean.

All the following macros are in meters The vertical position of the camera

#### 6.5.1.10 `#define LEAP_POINTER_SIZE 1.0f`

The size of the [LeapmotionPointer](#).

#### 6.5.1.11 `#define MODULE_SIZE 0.12f`

The size of a a half of a Roombots module.

#### 6.5.1.12 `#define PINCHING_LIMIT 0.7f`

The minimal pinching value to consider that the hand is pinching.

#### 6.5.1.13 `#define ROOM_SIZE 5.0f`

The size of the room.

#### 6.5.1.14 `#define TRASH_CAN_SIZE 2.0f`

The size of the [TrashCan](#).

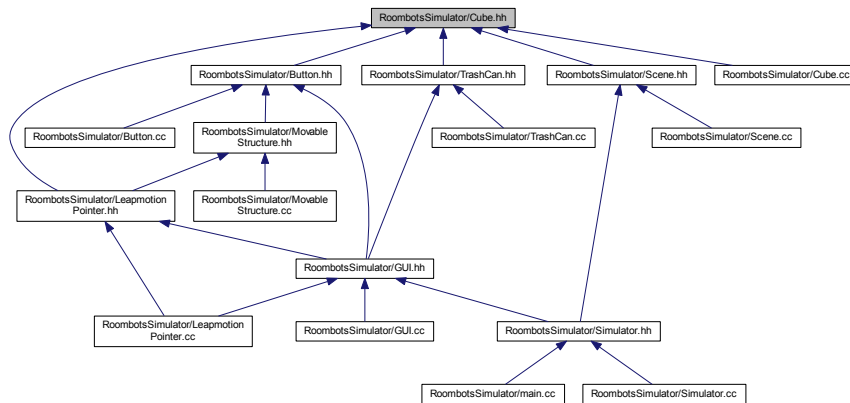
## 6.6 RoombotsSimulator/Cube.cc File Reference

```
#include "Cube.hh"
```

## 6.7 RoombotsSimulator/Cube.hh File Reference

```
#include "Model.hh"
```

This graph shows which files directly or indirectly include this file:



### Classes

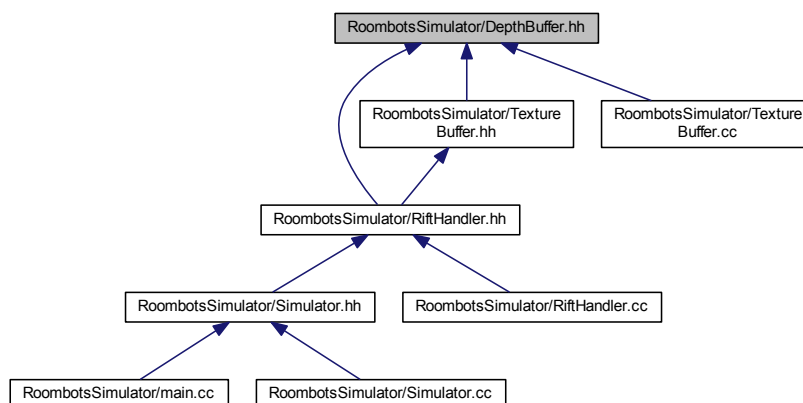
- class [Cube](#)

## 6.8 RoombotsSimulator/DepthBuffer.cc File Reference

## 6.9 RoombotsSimulator/DepthBuffer.hh File Reference

```
#include "common.hh"
```

This graph shows which files directly or indirectly include this file:



## Classes

- struct [DepthBuffer](#)

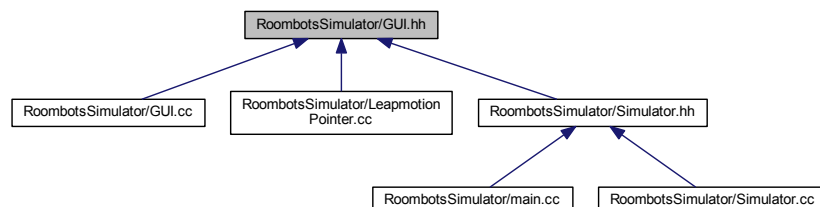
## 6.10 RoombotsSimulator/GUI.cc File Reference

```
#include "GUI.hh"
```

## 6.11 RoombotsSimulator/GUI.hh File Reference

```
#include "Button.hh"  
#include "LeapmotionPointer.hh"  
#include "Structure.hh"  
#include "Position.hh"  
#include "TrashCan.hh"
```

This graph shows which files directly or indirectly include this file:



## Classes

- class [GUI](#)

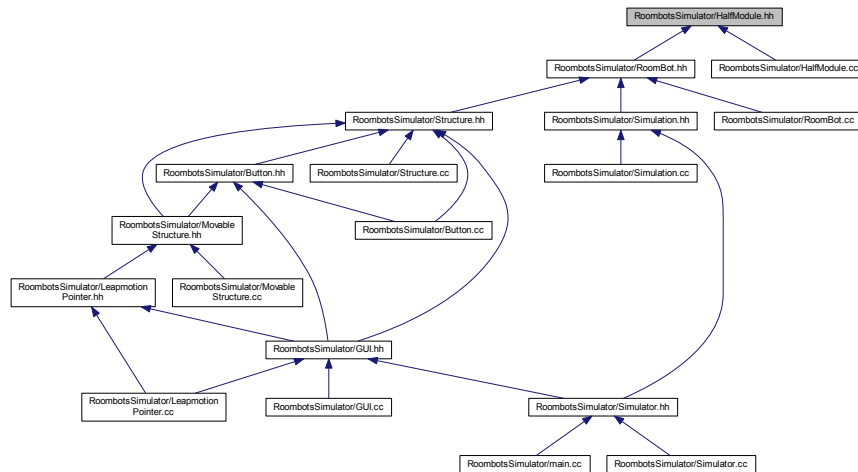
## 6.12 RoombotsSimulator/HalfModule.cc File Reference

```
#include "HalfModule.hh"  
#include "Position.hh"
```

## 6.13 RoombotsSimulator/HalfModule.hh File Reference

```
#include "OBJModel.hh"
#include "Position.hh"
```

This graph shows which files directly or indirectly include this file:



### Classes

- class [HalfModule](#)

## 6.14 RoombotsSimulator/LeapmotionPointer.cc File Reference

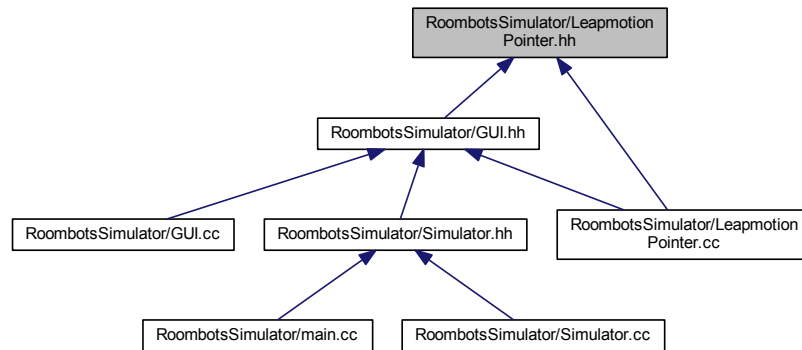
```
#include "LeapmotionPointer.hh"
#include "GUI.hh"
```

## 6.15 RoombotsSimulator/LeapmotionPointer.hh File Reference

```
#include "common.hh"
#include "Cube.hh"
#include "Quad.hh"
#include "MovableStructure.hh"
```



This graph shows which files directly or indirectly include this file:



## Classes

- class [LeapmotionPointer](#)

## 6.16 RoombotsSimulator/main.cc File Reference

```
#include "Simulator.hh"
```

## Functions

- void [display](#) ()
- void [renderScene](#) ()
- void [handleKeyboard](#) (unsigned char key, int x, int y)
- void [resize](#) (int w, int h)
- void [close](#) ()
- int [main](#) (int argc, char \*\*argv)

### 6.16.1 Function Documentation

6.16.1.1 void [close](#) ( )

6.16.1.2 void [display](#) ( )

## Author

Valentin NIGOLIAN [valentin.nigolian@epfl.ch](mailto:valentin.nigolian@epfl.ch) Fall 2015



## 6.19.1 Function Documentation

### 6.19.1.1 `__declspec ( align(16) )`

This class represents an object that can be drawn in an OpenGL context. All objects in this software that must be drawn must have at least one Model. It uses '`__declspec(align(16))`' to be aligned on the heap

IMPORTANT NOTE : all derived classes MUST call 'Init()' in their constructor in order to initialize them properly This constructor initializes all the parts of the Model

- `vShaderFileName` The name of the file containing the vertex shader
- `fShaderFileName` The name of the file containing the fragment shader
- `textureFileName` The name of the file containing the texture
- `color` The RGBA-formatted color of the model (if no texture is used)

This constructor is an alias of the first one that uses 'std::string' instead of 'const char\*'

This is necessary when using '`__declspec`'

Same as above

Sets the model matrix

- `M` The model matrix defines the scale, rotation and translation of the model It doesn't change its vertices or definition but how and where it will appear in the scene

Draws the model into the scene.

- `VP` The Projection-View matrix, as required by the shaders

Draws the Model twice. Once normally and once only with the lines

Cleans up everything that has been set up during initialization :

- the VBO
- the VAO
- the shading program
- the texture

The two following methods are pure virtual as they are used to define the shape and texture coordinates of the model that has to be rendered and thus have to be defined for every model The vectors passed in argument in both methods are created in the 'Init' method IMPORTANT : both vectors MUST have the same size as OpenGL makes a correspondance between the elements of those vectors

<The model matrix

<Set on 'true' once initialized

< The vertex shader's file name

< The fragment shader's file name

< The texture's file name

Calls 'SetVertices' and 'SetUVs' and makes sure both have the same size

Defines the vertices of the model. Each series of 3 vertices (each stored as a `vec3`) will compose a new triangle to be rendered. If the number of vertices isn't a multiple of three, the exceeding vertices will simply be ignored.

Defines the texture coordinates (also called 'UVs') of the model UVs are used to make a correspondance between a vertex and a position in the texture in order to determine the color of every vertex and interpolate between them to set the color of every pixel

Initializes everything. (called from the construtor)

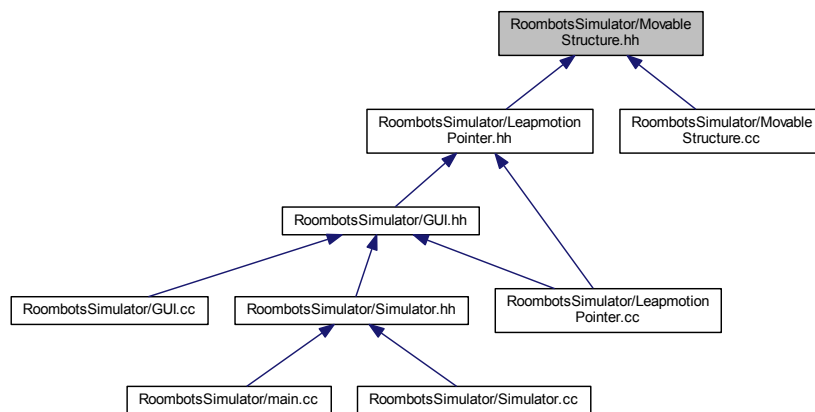
## 6.20 RoombotsSimulator/MovableStructure.cc File Reference

```
#include "MovableStructure.hh"
```

## 6.21 RoombotsSimulator/MovableStructure.hh File Reference

```
#include "Button.hh"
#include "Structure.hh"
```

This graph shows which files directly or indirectly include this file:



### Classes

- class [MovableStructure](#)

## 6.22 RoombotsSimulator/OBJModel.cc File Reference

```
#include "OBJModel.hh"
```

## 6.23 RoombotsSimulator/OBJModel.hh File Reference

```
#include "Model.hh"
#include <string>
```

- class OBJModel

```
#include "common.hh"
#include "Position.hh"
```

```
graph BT; RoombotsSimulatorPathFinder[RoombotsSimulator/PathFinder.hh] --> RoombotsSimulatorBrutePathFinder[RoombotsSimulator/BrutePathFinder.hh]; RoombotsSimulatorPathFinder --> RoombotsSimulatorSimulation[RoombotsSimulator/Simulation.hh]; RoombotsSimulatorBrutePathFinder --> RoombotsSimulatorBrutePathFinderCC[RoombotsSimulator/BrutePathFinder.cc]; RoombotsSimulatorSimulation --> RoombotsSimulatorBrutePathFinderCC; RoombotsSimulatorSimulation --> RoombotsSimulatorSimulationCC[RoombotsSimulator/Simulation.cc]; RoombotsSimulatorSimulation --> RoombotsSimulatorSimulator[RoombotsSimulator/Simulator.hh]; RoombotsSimulatorSimulator --> RoombotsSimulatorMain[RoombotsSimulator/main.cc]; RoombotsSimulatorSimulator --> RoombotsSimulatorSimulatorCC[RoombotsSimulator/Simulator.cc];
```

- class **PathFinder**

## Typedefs

- typedef std::vector< [Position](#) > [Path](#)

### 6.24.1 Typedef Documentation

#### 6.24.1.1 typedef std::vector<[Position](#)> [Path](#)

#### Author

Valentin NIGOLIAN [valentin.nigolian@epfl.ch](mailto:valentin.nigolian@epfl.ch) Fall 2015

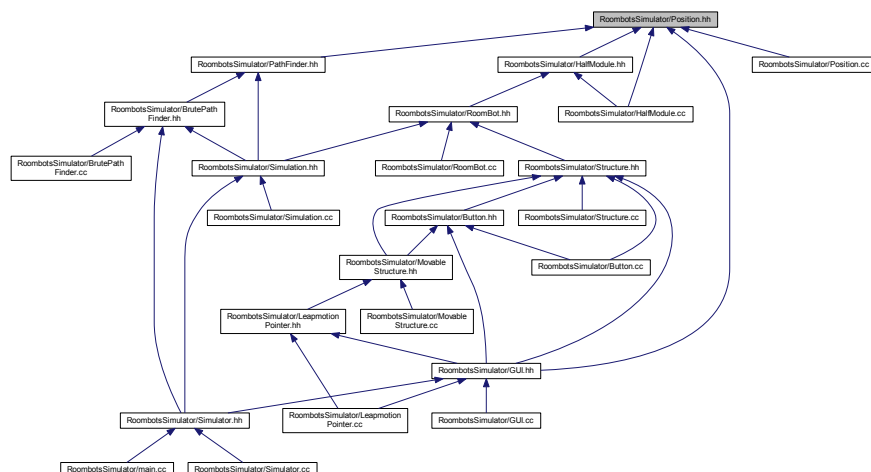
## 6.25 RoombotsSimulator/Position.cc File Reference

```
#include "Position.hh"
```

## 6.26 RoombotsSimulator/Position.hh File Reference

```
#include "common.hh"
#include "glm\glm.hpp"
#include "glm\gtc\matrix_transform.hpp"
#include "glm\gtx\transform.hpp"
#include "glm\gtx\euler_angles.hpp"
```

This graph shows which files directly or indirectly include this file:



## Classes

- class [Position](#)

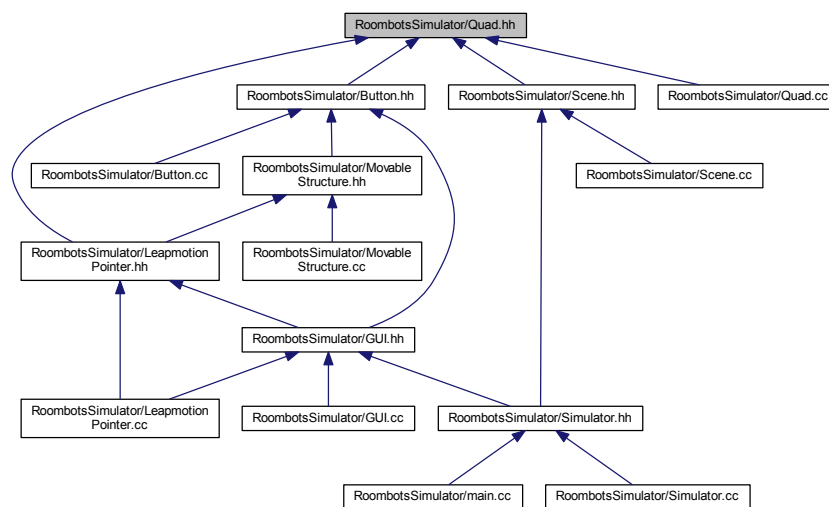
## 6.27 RoombotsSimulator/Quad.cc File Reference

```
#include "Quad.hh"
```

## 6.28 RoombotsSimulator/Quad.hh File Reference

```
#include "Model.hh"
```

This graph shows which files directly or indirectly include this file:



### Classes

- class [Quad](#)

## 6.29 RoombotsSimulator/RiftHandler.cc File Reference

```
#include "RiftHandler.hh"
```

### Functions

- glm::mat4 [OVR\\_Mat4\\_to\\_GLM\\_mat4](#) (OVR::Matrix4f sourceMatrix)

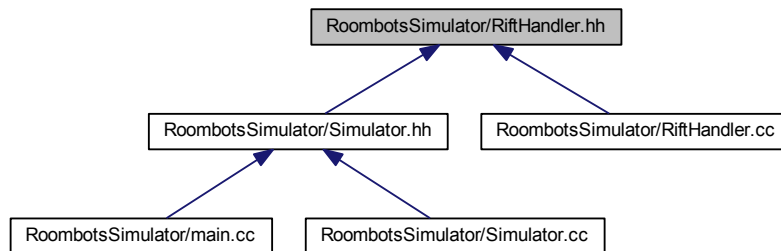
### 6.29.1 Function Documentation

6.29.1.1 `glm::mat4 OVR_Mat4_to_GLM_mat4 ( OVR::Matrix4f sourceMatrix )`

## 6.30 RoombotsSimulator/RiftHandler.hh File Reference

```
#include "common.hh"
#include "DepthBuffer.hh"
#include "TextureBuffer.hh"
```

This graph shows which files directly or indirectly include this file:



### Classes

- class [RiftHandler](#)

### Typedefs

- typedef `void(* DisplayFunction) ()`

### 6.30.1 Typedef Documentation

6.30.1.1 `typedef void(* DisplayFunction) ()`

This class makes the Rift much simpler to use. All that must be done is first initialise the Rift with 'Init(DisplayFunction)', where 'DisplayFunction' is the function that draws what we want to display on the Rift and then call 'DisplayOnRift' when we want to display something on the Rift. To take its orientation into account when drawing the scene (in the DisplayFunction passed in argument in 'Init()'), one must can use the 'ovrViewProjMatrix()' or 'glmViewProjMatrix()' methods to get the view-projection matrix in the form of a `OVR::Matrix4f` or a `glm::mat4` matrix. The `OVR::Matrix4f` is how it is created by the Rift's SDK but the `glm::mat4` one is easier to use with OpenGL.

## 6.31 RoombotsSimulator/RoomBot.cc File Reference

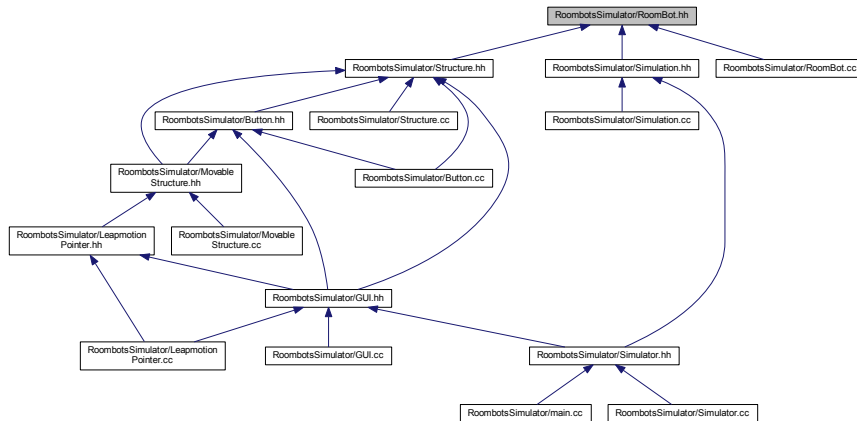
```
#include "RoomBot.hh"
```



## 6.32 RoombotsSimulator/RoomBot.hh File Reference

```
#include "HalfModule.hh"
```

This graph shows which files directly or indirectly include this file:



### Classes

- class [RoomBot](#)

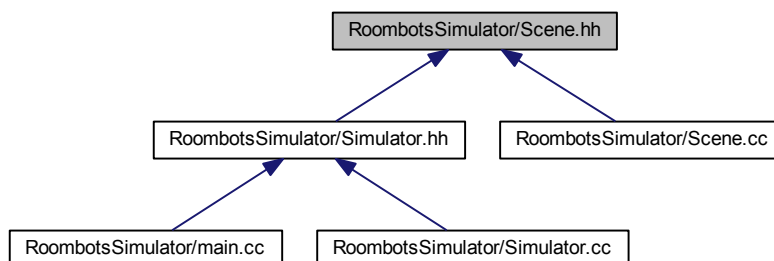
## 6.33 RoombotsSimulator/Scene.cc File Reference

```
#include "Scene.hh"
```

## 6.34 RoombotsSimulator/Scene.hh File Reference

```
#include "common.hh"
#include "Model.hh"
#include "Quad.hh"
#include "Cube.hh"
```

This graph shows which files directly or indirectly include this file:

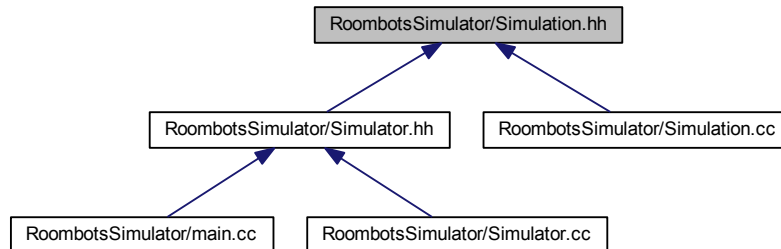




## 6.38 RoombotsSimulator/Simulation.hh File Reference

```
#include "RoomBot.hh"
#include "BrutePathFinder.hh"
#include "PathFinder.hh"
```

This graph shows which files directly or indirectly include this file:



### Classes

- class [Simulation](#)

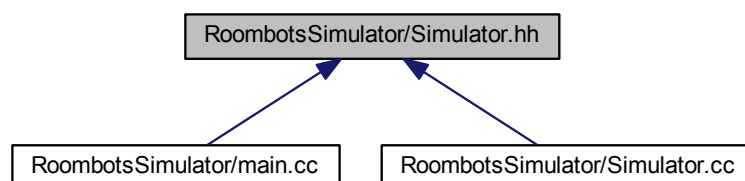
## 6.39 RoombotsSimulator/Simulator.cc File Reference

```
#include "Simulator.hh"
```

## 6.40 RoombotsSimulator/Simulator.hh File Reference

```
#include "common.hh"
#include "RiftHandler.hh"
#include "Scene.hh"
#include "GUI.hh"
#include "OBJModel.hh"
#include "BrutePathFinder.hh"
#include "Simulation.hh"
```

This graph shows which files directly or indirectly include this file:



## Classes

- class [Simulator](#)

## 6.41 RoombotsSimulator/Structure.cc File Reference

```
#include "Structure.hh"
```

## Functions

- int [min](#) (int a, int b)
- int [min](#) (int a, int b, int c)

### 6.41.1 Function Documentation

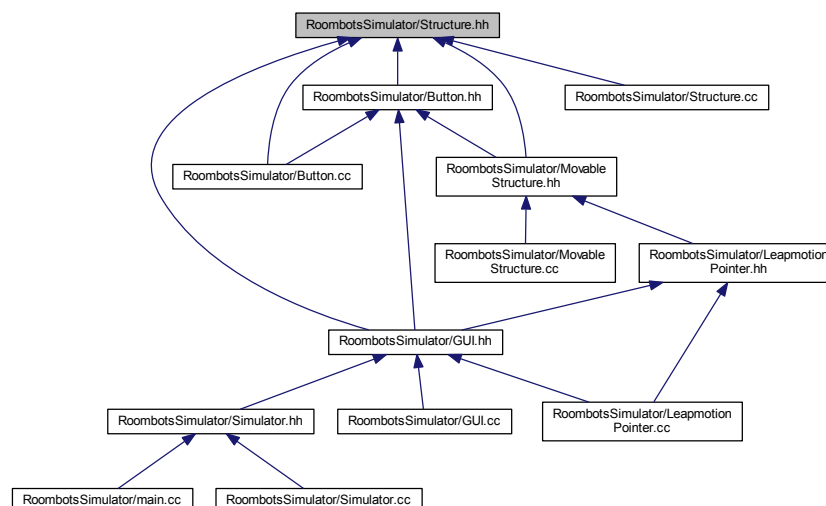
6.41.1.1 int min ( int a, int b )

6.41.1.2 int min ( int a, int b, int c )

## 6.42 RoombotsSimulator/Structure.hh File Reference

```
#include "RoomBot.hh"
```

This graph shows which files directly or indirectly include this file:



## Classes

- class [Structure](#)

## 6.43 RoombotsSimulator/TextureBuffer.cc File Reference

```
#include "common.hh"  
#include "DepthBuffer.hh"
```

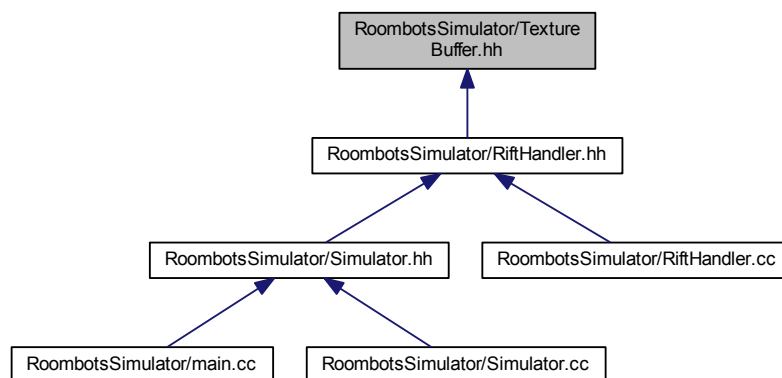
### Classes

- struct [TextureBuffer](#)

## 6.44 RoombotsSimulator/TextureBuffer.hh File Reference

```
#include "common.hh"  
#include "DepthBuffer.hh"
```

This graph shows which files directly or indirectly include this file:



### Classes

- struct [TextureBuffer](#)

## 6.45 RoombotsSimulator/TrashCan.cc File Reference

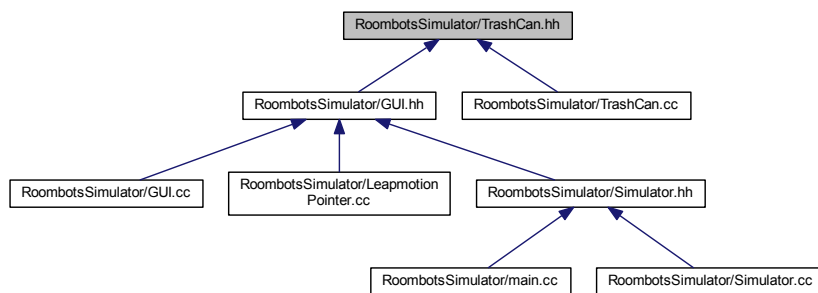
```
#include "TrashCan.hh"
```

## 6.46 RoombotsSimulator/TrashCan.hh File Reference

```
#include "common.hh"
```

```
#include "Cube.hh"
```

This graph shows which files directly or indirectly include this file:



### Classes

- class [TrashCan](#)

# Index

- `__declspec`
    - `Model.hh`, [67](#)
  - `_objfilename`
    - `OBJModel`, [30](#)
  - `~Simulator`
    - `Simulator`, [48](#)
- `AdaptToMode`
  - `LeapmotionPointer`, [22](#)
- `AddButton`
  - `GUI`, [16](#)
- `AddModel`
  - `Scene`, [41](#)
- `AssignStructure`
  - `LeapmotionPointer`, [23](#)
- `AssignedStructure`
  - `Button`, [12](#)
  - `LeapmotionPointer`, [22](#)
- `BOX_COORDINATE_SYSTEM_SCALE_CONVERSION`
  - `common.hh`, [60](#)
- `BUTTON_DEPTH_OFFSET`
  - `common.hh`, [60](#)
- `BUTTON_RIGHT_START`
  - `common.hh`, [60](#)
- `BUTTON_SEPARATION`
  - `common.hh`, [60](#)
- `BUTTON_SIZE`
  - `common.hh`, [60](#)
- `BUTTON_UP_START`
  - `common.hh`, [60](#)
- `Backwards`
  - `Simulator`, [48](#)
- `BrutePathFinder`, [9](#)
  - `Name`, [10](#)
  - `Run`, [10](#)
- `Button`, [11](#)
  - `AssignedStructure`, [12](#)
  - `Button`, [12](#)
  - `CleanUp`, [12](#)
  - `d_ID`, [12](#)
  - `d_model`, [12](#)
  - `d_p_structure`, [12](#)
  - `d_position`, [13](#)
  - `d_shadow`, [13](#)
  - `Draw`, [12](#)
  - `ID`, [12](#)
  - `Position`, [12](#)
- `COORDINATE_SYSTEM_SCALE_CONVERSION`
  - `common.hh`, [60](#)
- `CenterOffset`
  - `Structure`, [52](#)
- `CheckForPinchedStructure`
  - `GUI`, [16](#)
- `CleanUp`
  - `Button`, [12](#)
  - `GUI`, [16](#)
  - `HalfModule`, [20](#)
  - `LeapmotionPointer`, [23](#)
  - `RiftHandler`, [36](#)
  - `Scene`, [41](#)
  - `Simulator`, [48](#)
  - `TrashCan`, [56](#)
- `Close`
  - `Simulator`, [48](#)
- `close`
  - `main.cc`, [65](#)
- `CloseEnough`
  - `MovableStructure`, [26](#)
- `common.hh`
  - `BOX_COORDINATE_SYSTEM_SCALE_CONVERSION`, [60](#)
  - `BUTTON_DEPTH_OFFSET`, [60](#)
  - `BUTTON_RIGHT_START`, [60](#)
  - `BUTTON_SEPARATION`, [60](#)
  - `BUTTON_SIZE`, [60](#)
  - `BUTTON_UP_START`, [60](#)
  - `COORDINATE_SYSTEM_SCALE_CONVERSION`, [60](#)
  - `DRAW_RADIUS`, [60](#)
  - `EYES_POSITION`, [61](#)
  - `LEAP_POINTER_SIZE`, [61](#)
  - `MODULE_SIZE`, [61](#)
  - `PINCHING_LIMIT`, [61](#)
  - `ROOM_SIZE`, [61](#)
  - `TRASH_CAN_SIZE`, [61](#)
- `CreateProgram`
  - `ShaderLoader`, [43](#)
- `CreateShader`
  - `ShaderLoader`, [43](#)
- `Cube`, [13](#)
  - `Cube`, [14](#)
  - `SetUVs`, [14](#)
  - `SetVertices`, [14](#)
- `d_EyeRenderDesc`
  - `RiftHandler`, [37](#)
- `d_GUI`

- Simulator, [50](#)
- d\_ID
  - Button, [12](#)
  - MovableStructure, [27](#)
- d\_buttonID
  - MovableStructure, [27](#)
- d\_buttons
  - GUI, [18](#)
- d\_centerOffset
  - Structure, [53](#)
- d\_controller
  - LeapmotionPointer, [24](#)
- d\_currentStep
  - Simulation, [46](#)
- d\_displayFunction
  - RiftHandler, [37](#)
- d\_eyeDepthBuffer
  - RiftHandler, [37](#)
- d\_eyeRenderTexture
  - RiftHandler, [37](#)
- d\_filename
  - Structure, [53](#)
- d\_halfModuleA
  - RoomBot, [40](#)
- d\_halfModuleB
  - RoomBot, [40](#)
- d\_halfModules
  - Simulation, [46](#)
- d\_height
  - Simulator, [50](#)
- d\_hmd
  - RiftHandler, [37](#)
- d\_init
  - LeapmotionPointer, [24](#)
  - Simulation, [46](#)
- d\_instance
  - Simulator, [50](#)
- d\_invertedWorldMatrix
  - LeapmotionPointer, [24](#)
- d\_isVisible
  - RiftHandler, [37](#)
- d\_mirrorFBO
  - RiftHandler, [37](#)
- d\_mirrorTexture
  - RiftHandler, [37](#)
- d\_mode
  - Simulator, [50](#)
- d\_model
  - Button, [12](#)
  - TrashCan, [56](#)
- d\_models
  - Scene, [42](#)
- d\_moving
  - MovableStructure, [28](#)
- d\_nButtons
  - GUI, [18](#)
- d\_nModels
  - Scene, [42](#)
- d\_nStructures
  - GUI, [18](#)
- d\_over
  - Simulation, [46](#)
- d\_p\_circle
  - GUI, [18](#)
  - HalfModule, [21](#)
- d\_p\_gui
  - LeapmotionPointer, [24](#)
- d\_p\_hemi1
  - GUI, [18](#)
- d\_p\_hemi2
  - GUI, [18](#)
- d\_p\_hemisphere1
  - HalfModule, [21](#)
- d\_p\_hemisphere2
  - HalfModule, [21](#)
- d\_p\_pointerModel
  - LeapmotionPointer, [24](#)
- d\_p\_referencePointerModel
  - LeapmotionPointer, [24](#)
- d\_p\_shadow
  - LeapmotionPointer, [24](#)
- d\_p\_structure
  - Button, [12](#)
  - LeapmotionPointer, [24](#)
  - MovableStructure, [28](#)
- d\_pathFinder
  - Simulator, [50](#)
- d\_paths
  - Simulation, [46](#)
- d\_pointer
  - GUI, [18](#)
- d\_position
  - Button, [13](#)
  - HalfModule, [21](#)
  - LeapmotionPointer, [24](#)
  - MovableStructure, [28](#)
  - TrashCan, [56](#)
- d\_refClock
  - Simulation, [46](#)
- d\_rift
  - Simulator, [50](#)
- d\_roof
  - Scene, [42](#)
- d\_roomBots
  - Structure, [53](#)
- d\_running
  - Simulator, [50](#)
- d\_scene
  - Simulator, [50](#)
- d\_shadow
  - Button, [13](#)
- d\_simulation
  - Simulator, [50](#)
- d\_structures
  - GUI, [18](#)
- d\_trackingState



- RiftHandler, 38
- d\_trashCan
  - GUI, 19
- d\_viewProjMatrix
  - RiftHandler, 38
- d\_width
  - Simulator, 51
- d\_windowID
  - Simulator, 51
- d\_worldMatrix
  - Simulator, 51
- d\_x
  - Position, 33
- d\_y
  - Position, 33
- d\_z
  - Position, 33
- DRAG\_RADIUS
  - common.hh, 60
- DefaultFragmentShader
  - ShaderLoader, 43
- DefaultVertexShader
  - ShaderLoader, 43
- DepthBuffer, 14
  - DepthBuffer, 15
  - texId, 15
- Display
  - Simulator, 48
- display
  - main.cc, 65
- DisplayFunction
  - RiftHandler.hh, 72
- DisplayOnRift
  - RiftHandler, 36
- distanceTo
  - Position, 32
- Drag
  - MovableStructure, 26
- Draw
  - Button, 12
  - HalfModule, 20
  - LeapmotionPointer, 23
  - MovableStructure, 27
  - RoomBot, 39
  - Simulation, 45
  - Structure, 52
  - TrashCan, 56
- Drop
  - MovableStructure, 27
- DroppedStructure
  - GUI, 17
- EYES\_POSITION
  - common.hh, 61
- fbold
  - TextureBuffer, 54
- Forward
  - Simulator, 48
- GUI, 15
  - AddButton, 16
  - CheckForPinchedStructure, 16
  - CleanUp, 16
  - d\_buttons, 18
  - d\_nButtons, 18
  - d\_nStructures, 18
  - d\_p\_circle, 18
  - d\_p\_hemi1, 18
  - d\_p\_hemi2, 18
  - d\_pointer, 18
  - d\_structures, 18
  - d\_trashCan, 19
  - DroppedStructure, 17
  - GetAllRoombotsPositions, 17
  - Init, 17
  - NButtons, 17
  - PopStructure, 17
  - Render, 17
  - Update, 17
  - UpdatePointer, 17
  - UpdateWorldMatrix, 18
- GetAllRoombotsPositions
  - GUI, 17
- GetPosition
  - HalfModule, 20
  - MovableStructure, 27
- GetSize
  - TextureBuffer, 54
- glmViewProjMatrix
  - RiftHandler, 36
- HalfModule, 19
  - CleanUp, 20
  - d\_p\_circle, 21
  - d\_p\_hemisphere1, 21
  - d\_p\_hemisphere2, 21
  - d\_position, 21
  - Draw, 20
  - GetPosition, 20
  - HalfModule, 20
  - SetPosition, 20
- HandleKeyboard
  - Simulator, 48
- handleKeyboard
  - main.cc, 65
- ID
  - Button, 12
- Init
  - GUI, 17
  - LeapmotionPointer, 23
  - RiftHandler, 36
  - Scene, 41
  - Simulator, 48
- InitRift
  - Simulator, 49
- InitSimulation
  - Simulator, 49

- Initialize
  - Simulation, 45
- Instance
  - Simulator, 49
- IsInitialized
  - Simulation, 45
- IsOver
  - Simulation, 45
- LEAP\_POINTER\_SIZE
  - common.hh, 61
- LeapmotionPointer, 21
  - AdaptToMode, 22
  - AssignStructure, 23
  - AssignedStructure, 22
  - CleanUp, 23
  - d\_controller, 24
  - d\_init, 24
  - d\_invertedWorldMatrix, 24
  - d\_p\_gui, 24
  - d\_p\_pointerModel, 24
  - d\_p\_referencePointerModel, 24
  - d\_p\_shadow, 24
  - d\_p\_structure, 24
  - d\_position, 24
  - Draw, 23
  - Init, 23
  - Pinching, 23
  - Position, 23
  - update, 23
  - UpdateWorldMatrix, 23
- Left
  - Simulator, 49
- LinkerButtonID
  - MovableStructure, 27
- MODULE\_SIZE
  - common.hh, 61
- main
  - main.cc, 66
- main.cc
  - close, 65
  - display, 65
  - handleKeyboard, 65
  - main, 66
  - renderScene, 66
  - resize, 66
- MainLoop
  - Simulator, 49
- MiddlePosition
  - RoomBot, 39
- min
  - Structure.cc, 76
- Model.hh
  - \_\_declspec, 67
- MovableStructure, 25
  - CloseEnough, 26
  - d\_ID, 27
  - d\_buttonID, 27
  - d\_moving, 28
  - d\_p\_structure, 28
  - d\_position, 28
  - Drag, 26
  - Draw, 27
  - Drop, 27
  - GetPosition, 27
  - LinkerButtonID, 27
  - MovableStructure, 26
  - RoombotsPositions, 27
  - SetCenterOffset, 27
  - SetPosition, 27
- NButtons
  - GUI, 17
- Name
  - BrutePathFinder, 10
  - PathFinder, 31
- NextStep
  - Simulation, 45
- OBJModel, 28
  - \_objfilename, 30
  - OBJModel, 29
  - SetUVs, 30
  - SetVertices, 30
- OVR\_Mat4\_to\_GLM\_mat4
  - RiftHandler.cc, 72
- operator!=
  - Position, 32
- operator\*
  - Position, 33
- operator\*=
  - Position, 33
- operator+
  - Position, 33
- operator+=
  - Position, 33
- operator-
  - Position, 33
- operator-=
  - Position, 33
- operator==
  - Position, 33
- ovrViewProjMatrix
  - RiftHandler, 36
- PINCHING\_LIMIT
  - common.hh, 61
- Path
  - PathFinder.hh, 70
- PathFinder, 30
  - Name, 31
  - Run, 31
- PathFinder.hh
  - Path, 70
- Pinching
  - LeapmotionPointer, 23
- PopStructure

- GUI, [17](#)
- Position, [31](#)
  - Button, [12](#)
  - d\_x, [33](#)
  - d\_y, [33](#)
  - d\_z, [33](#)
  - distanceTo, [32](#)
  - LeapmotionPointer, [23](#)
  - operator!=, [32](#)
  - operator\*, [33](#)
  - operator\*=: [33](#)
  - operator+, [33](#)
  - operator+=, [33](#)
  - operator-, [33](#)
  - operator-=, [33](#)
  - operator==, [33](#)
  - Position, [32](#)
  - Print, [33](#)
  - ToGLM, [33](#)
  - TrashCan, [56](#)
  - x, [33](#)
  - y, [33](#)
  - z, [33](#)
- PositionA
  - RoomBot, [40](#)
- PositionB
  - RoomBot, [40](#)
- Print
  - Position, [33](#)
- Quad, [34](#)
  - Quad, [35](#)
  - SetUVs, [35](#)
  - SetVertices, [35](#)
- ROOM\_SIZE
  - common.hh, [61](#)
- ReadShader
  - ShaderLoader, [43](#)
- Render
  - GUI, [17](#)
  - Scene, [41](#)
- RenderScene
  - Simulator, [49](#)
- renderScene
  - main.cc, [66](#)
- Reset
  - Simulation, [45](#)
- Resize
  - Simulator, [49](#)
- resize
  - main.cc, [66](#)
- ResolutionHeight
  - RiftHandler, [37](#)
- ResolutionWidth
  - RiftHandler, [37](#)
- RiftHandler, [35](#)
  - CleanUp, [36](#)
  - d\_EyeRenderDesc, [37](#)
  - d\_displayFunction, [37](#)
  - d\_eyeDepthBuffer, [37](#)
  - d\_eyeRenderTexture, [37](#)
  - d\_hmd, [37](#)
  - d\_isVisible, [37](#)
  - d\_mirrorFBO, [37](#)
  - d\_mirrorTexture, [37](#)
  - d\_trackingState, [38](#)
  - d\_viewProjMatrix, [38](#)
  - DisplayOnRift, [36](#)
  - glmViewProjMatrix, [36](#)
  - Init, [36](#)
  - ovrViewProjMatrix, [36](#)
  - ResolutionHeight, [37](#)
  - ResolutionWidth, [37](#)
- RiftHandler.cc
  - OVR\_Mat4\_to\_GLM\_mat4, [72](#)
- RiftHandler.hh
  - DisplayFunction, [72](#)
- Right
  - Simulator, [49](#)
- RoomBot, [38](#)
  - d\_halfModuleA, [40](#)
  - d\_halfModuleB, [40](#)
  - Draw, [39](#)
  - MiddlePosition, [39](#)
  - PositionA, [40](#)
  - PositionB, [40](#)
  - RoomBot, [39](#)
- RoombotsPositions
  - MovableStructure, [27](#)
  - Structure, [52](#)
- RoombotsSimulator/BrutePathFinder.cc, [57](#)
- RoombotsSimulator/BrutePathFinder.hh, [57](#)
- RoombotsSimulator/Button.cc, [58](#)
- RoombotsSimulator/Button.hh, [58](#)
- RoombotsSimulator/Cube.cc, [61](#)
- RoombotsSimulator/Cube.hh, [62](#)
- RoombotsSimulator/DepthBuffer.cc, [62](#)
- RoombotsSimulator/DepthBuffer.hh, [62](#)
- RoombotsSimulator/GUI.cc, [63](#)
- RoombotsSimulator/GUI.hh, [63](#)
- RoombotsSimulator/HalfModule.cc, [63](#)
- RoombotsSimulator/HalfModule.hh, [64](#)
- RoombotsSimulator/LeapmotionPointer.cc, [64](#)
- RoombotsSimulator/LeapmotionPointer.hh, [64](#)
- RoombotsSimulator/Model.cc, [66](#)
- RoombotsSimulator/Model.hh, [66](#)
- RoombotsSimulator/MovableStructure.cc, [68](#)
- RoombotsSimulator/MovableStructure.hh, [68](#)
- RoombotsSimulator/ObjModel.cc, [68](#)
- RoombotsSimulator/ObjModel.hh, [68](#)
- RoombotsSimulator/PathFinder.hh, [69](#)
- RoombotsSimulator/Position.cc, [70](#)
- RoombotsSimulator/Position.hh, [70](#)
- RoombotsSimulator/Quad.cc, [71](#)
- RoombotsSimulator/Quad.hh, [71](#)
- RoombotsSimulator/RiftHandler.cc, [71](#)

- RoombotsSimulator/RiftHandler.hh, 72
- RoombotsSimulator/RoomBot.cc, 72
- RoombotsSimulator/RoomBot.hh, 73
- RoombotsSimulator/Scene.cc, 73
- RoombotsSimulator/Scene.hh, 73
- RoombotsSimulator/ShaderLoader.cc, 74
- RoombotsSimulator/ShaderLoader.hh, 74
- RoombotsSimulator/Simulation.cc, 74
- RoombotsSimulator/Simulation.hh, 75
- RoombotsSimulator/Simulator.cc, 75
- RoombotsSimulator/Simulator.hh, 75
- RoombotsSimulator/Structure.cc, 76
- RoombotsSimulator/Structure.hh, 76
- RoombotsSimulator/TextureBuffer.cc, 77
- RoombotsSimulator/TextureBuffer.hh, 77
- RoombotsSimulator/TrashCan.cc, 77
- RoombotsSimulator/TrashCan.hh, 78
- RoombotsSimulator/common.hh, 58
- RoombotsSimulator/main.cc, 65
- RoombotsSimulator/mainpage.dox, 66
- Run
  - BrutePathFinder, 10
  - PathFinder, 31
  - Simulation, 45
- Scene, 40
  - AddModel, 41
  - CleanUp, 41
  - d\_models, 42
  - d\_nModels, 42
  - d\_roof, 42
  - Init, 41
  - Render, 41
- SetAndClearRenderSurface
  - TextureBuffer, 54
- SetCenterOffset
  - MovableStructure, 27
  - Structure, 52
- SetPosition
  - HalfModule, 20
  - MovableStructure, 27
- SetUVs
  - Cube, 14
  - OBJModel, 30
  - Quad, 35
- SetVertices
  - Cube, 14
  - OBJModel, 30
  - Quad, 35
- ShaderLoader, 42
  - CreateProgram, 43
  - CreateShader, 43
  - DefaultFragmentShader, 43
  - DefaultVertexShader, 43
  - ReadShader, 43
- Simulation, 44
  - d\_currentStep, 46
  - d\_halfModules, 46
  - d\_init, 46
  - d\_over, 46
  - d\_paths, 46
  - d\_refClock, 46
  - Draw, 45
  - Initialize, 45
  - IsInitialized, 45
  - IsOver, 45
  - NextStep, 45
  - Reset, 45
  - Run, 45
- Simulator, 46
  - ~Simulator, 48
  - Backwards, 48
  - CleanUp, 48
  - Close, 48
  - d\_GUI, 50
  - d\_height, 50
  - d\_instance, 50
  - d\_mode, 50
  - d\_pathFinder, 50
  - d\_rift, 50
  - d\_running, 50
  - d\_scene, 50
  - d\_simulation, 50
  - d\_width, 51
  - d\_windowID, 51
  - d\_worldMatrix, 51
  - Display, 48
  - Forward, 48
  - HandleKeyboard, 48
  - Init, 48
  - InitRift, 49
  - InitSimulation, 49
  - Instance, 49
  - Left, 49
  - MainLoop, 49
  - RenderScene, 49
  - Resize, 49
  - Right, 49
  - Simulator, 48
  - Start, 49
  - SwitchViewMode, 49
  - WorldViewMatrix, 50
- Start
  - Simulator, 49
- Structure, 51
  - CenterOffset, 52
  - d\_centerOffset, 53
  - d\_filename, 53
  - d\_roomBots, 53
  - Draw, 52
  - RoombotsPositions, 52
  - SetCenterOffset, 52
  - Structure, 52
- Structure.cc
  - min, 76
- SwitchViewMode
  - Simulator, 49

- TRASH\_CAN\_SIZE
  - common.hh, [61](#)
- texId
  - DepthBuffer, [15](#)
  - TextureBuffer, [54](#)
- texSize
  - TextureBuffer, [54](#)
- TextureBuffer, [53](#)
  - fbold, [54](#)
  - GetSize, [54](#)
  - SetAndClearRenderSurface, [54](#)
  - texId, [54](#)
  - texSize, [54](#)
  - TextureBuffer, [54](#)
  - TextureSet, [54](#)
  - UnsetRenderSurface, [54](#)
- TextureSet
  - TextureBuffer, [54](#)
- ToGLM
  - Position, [33](#)
- TrashCan, [55](#)
  - CleanUp, [56](#)
  - d\_model, [56](#)
  - d\_position, [56](#)
  - Draw, [56](#)
  - Position, [56](#)
  - TrashCan, [55](#)
- UnsetRenderSurface
  - TextureBuffer, [54](#)
- Update
  - GUI, [17](#)
- update
  - LeapmotionPointer, [23](#)
- UpdatePointer
  - GUI, [17](#)
- UpdateWorldMatrix
  - GUI, [18](#)
  - LeapmotionPointer, [23](#)
- WorldViewMatrix
  - Simulator, [50](#)
- x
  - Position, [33](#)
- y
  - Position, [33](#)
- z
  - Position, [33](#)