

Necrodoggiecon and Cerberus

Generated on Fri May 27 2022 11:49:22 for Necrodoggiecon and Cerberus by Doxygen 1.9.4

Fri May 27 2022 11:49:22

1 Necrodoggiecon	1
1.1 How the project works	1
1.2 Instructions	1
1.2.1 How to compile the Engine	1
1.2.2 How to compile the Game	1
1.2.3 How to play the Game	1
1.2.3.1 Controls:	1
1.2.4 Naming Convention	1
1.2.4.1 Variables:	1
1.2.4.2 Functions:	2
1.2.4.3 Enums, Defines:	2
1.2.5 Links to the aspects of the Engine/Game	2
2 AI	3
2.1 Navigation and Pathfinding	3
2.2 Perception	3
2.3 Decision Making using a Finite State Machine	3
2.4 Enemies	4
2.4.1 AlarmEnemy	4
2.4.2 DogEnemy	4
2.4.3 GruntEnemy	5
2.5 Relating Classes:	5
3 Asset Manager	7
3.1 Managing Assets within the engine.	7
3.2 Relating Classes:	7
4 Audio	9
4.1 Adding, Playing and Managing Audio	9
4.2 Audio Emitters	9
4.3 Architecture	10
4.4 Relating Classes:	10
5 Utility	11
5.1 Relating Classes:	11
6 Weapon System	13
6.1 Strategy Design Pattern	13
6.2 Why does this Design Pattern work	13
6.3 Weapons	14
6.3.1 Melee Weapons	14
6.3.2 Range Weapons	15
6.4 Relating Classes:	15

7 Hierarchical Index	17
7.1 Class Hierarchy	17
8 Class Index	21
8.1 Class List	21
9 File Index	25
9.1 File List	25
10 Class Documentation	31
10.1 _Material Struct Reference	31
10.2 AlarmEnemy Class Reference	31
10.2.1 Detailed Description	32
10.2.2 Member Function Documentation	32
10.2.2.1 ChasePlayer()	32
10.2.2.2 OnDeath()	32
10.2.2.3 OnHit()	33
10.2.2.4 Update()	33
10.3 AssetManager Class Reference	33
10.3.1 Member Function Documentation	34
10.3.1.1 AddAudio()	34
10.3.1.2 AddMesh()	34
10.3.1.3 GetAudio()	34
10.3.1.4 GetDefaultMesh()	35
10.3.1.5 GetMesh()	35
10.3.1.6 GetTexture()	35
10.3.1.7 GetTextureWIC()	36
10.3.1.8 RemoveAudio()	36
10.4 AttackState Class Reference	36
10.4.1 Detailed Description	37
10.4.2 Member Function Documentation	37
10.4.2.1 Enter()	37
10.4.2.2 Exit()	37
10.4.2.3 Update()	38
10.5 AudioController Class Reference	38
10.5.1 Member Function Documentation	38
10.5.1.1 AddEmitter()	38
10.5.1.2 AddListener()	39
10.5.1.3 DestroyAudio()	39
10.5.1.4 GetAllEmittersWithinRange()	39
10.5.1.5 LoadAudio()	40
10.5.1.6 PlayAudio() [1/2]	40
10.5.1.7 PlayAudio() [2/2]	41

10.5.1.8 RemoveEmitter()	41
10.5.1.9 StopAudio()	41
10.6 AudioEmitterEntity Class Reference	42
10.6.1 Member Function Documentation	43
10.6.1.1 Load()	43
10.6.1.2 PlayAudio() [1/3]	43
10.6.1.3 PlayAudio() [2/3]	43
10.6.1.4 PlayAudio() [3/3]	44
10.6.1.5 SetAudio() [1/2]	44
10.6.1.6 SetAudio() [2/2]	44
10.6.1.7 SetRange()	45
10.6.1.8 Update()	45
10.7 CAIController Class Reference	45
10.7.1 Detailed Description	48
10.7.2 Member Function Documentation	48
10.7.2.1 ApplyDamage() [1/2]	48
10.7.2.2 ApplyDamage() [2/2]	48
10.7.2.3 AttackEnter()	49
10.7.2.4 AttackPlayer()	49
10.7.2.5 CanSee()	49
10.7.2.6 ChaseEnter()	49
10.7.2.7 ChasePlayer()	50
10.7.2.8 CollisionAvoidance()	50
10.7.2.9 HasCollided()	50
10.7.2.10 Investigating()	50
10.7.2.11 Movement()	51
10.7.2.12 Seek()	51
10.7.2.13 SetCurrentState()	51
10.7.2.14 SetPath()	51
10.7.2.15 SetPathNodes()	52
10.7.2.16 Update()	52
10.8 CameraManager Class Reference	52
10.8.1 Member Function Documentation	53
10.8.1.1 AddCamera()	53
10.8.1.2 GetAllCameras()	53
10.8.1.3 GetRenderingCamera()	53
10.8.1.4 RemoveCamera()	54
10.8.1.5 SetRenderingCamera()	54
10.9 CAnimationSpriteComponent Class Reference	54
10.9.1 Detailed Description	55
10.9.2 Member Function Documentation	55
10.9.2.1 SetAnimationRectPosition()	55

10.9.2.2 SetAnimationRectSize()	55
10.9.2.3 Update()	56
10.10 CAudio Class Reference	56
10.11 CAudioEmitterComponent Class Reference	56
10.11.1 Member Function Documentation	57
10.11.1.1 Draw()	57
10.11.1.2 Load() [1/2]	57
10.11.1.3 Load() [2/2]	58
10.11.1.4 Play()	58
10.11.1.5 SetRange()	58
10.11.1.6 Update()	59
10.12 CCamera Class Reference	59
10.12.1 Member Function Documentation	59
10.12.1.1 Update()	59
10.13 CCameraComponent Class Reference	60
10.13.1 Member Function Documentation	60
10.13.1.1 Draw()	61
10.13.1.2 getAttachedToParent()	61
10.13.1.3 GetPosition()	61
10.13.1.4 GetProjectionMatrix()	61
10.13.1.5 GetViewMatrix()	62
10.13.1.6 GetZoomLevel()	62
10.13.1.7 SetAttachedToParent()	62
10.13.1.8 SetZoomLevel()	62
10.13.1.9 Update()	63
10.14 CCharacter Class Reference	63
10.14.1 Member Function Documentation	64
10.14.1.1 ApplyDamage()	64
10.14.1.2 Update()	64
10.15 CComponent Class Reference	65
10.15.1 Detailed Description	66
10.15.2 Member Function Documentation	66
10.15.2.1 Draw()	66
10.15.2.2 GetTransform()	66
10.15.2.3 SetAnchor()	67
10.15.2.4 SetUseTranslucency()	67
10.15.2.5 Update()	67
10.16 CellData Struct Reference	67
10.17 CEmitter Class Reference	68
10.18 CEntity Class Reference	68
10.18.1 Detailed Description	69
10.18.2 Member Function Documentation	69

10.18.2.1 HasCollided()	69
10.18.2.2 SetIsUI()	70
10.18.2.3 Update()	70
10.19 CGridCursor Class Reference	70
10.19.1 Member Function Documentation	71
10.19.1.1 Update()	71
10.20 ChaseState Class Reference	71
10.20.1 Detailed Description	72
10.20.2 Member Function Documentation	72
10.20.2.1 Enter()	72
10.20.2.2 Exit()	72
10.20.2.3 Update()	72
10.21 CInteractable Class Reference	73
10.21.1 Member Function Documentation	73
10.21.1.1 GetLastCollidedObject()	74
10.21.1.2 GetSprite()	74
10.21.1.3 HasCollided()	74
10.21.1.4 OnInteract()	74
10.21.1.5 SetInteractRange()	75
10.21.1.6 SetTexture()	75
10.21.1.7 SetTextureWIC()	75
10.21.1.8 Update()	75
10.22 CMaterial Struct Reference	76
10.22.1 Detailed Description	76
10.23 CMesh Struct Reference	76
10.23.1 Detailed Description	77
10.24 CollisionComponent Class Reference	77
10.24.1 Member Function Documentation	77
10.24.1.1 Resolve()	77
10.25 ConstantBuffer Struct Reference	78
10.26 CParticle Class Reference	78
10.26.1 Member Function Documentation	79
10.26.1.1 Draw()	79
10.26.1.2 GetDirection()	79
10.26.1.3 GetLifetime()	80
10.26.1.4 getSpriteComponent()	80
10.26.1.5 GetVelocity()	80
10.26.1.6 SetDirection()	80
10.26.1.7 SetLifetime()	81
10.26.1.8 SetVelocity()	81
10.26.1.9 Update()	81
10.27 CParticleEmitter Class Reference	81

10.27.1 Member Function Documentation	82
10.27.1.1 Draw()	83
10.27.1.2 GetDirection()	83
10.27.1.3 GetLifetime()	83
10.27.1.4 GetVelocity()	84
10.27.1.5 SetDirection()	84
10.27.1.6 SetLifetime()	84
10.27.1.7 SetSize()	84
10.27.1.8 SetTexture()	85
10.27.1.9 SetVelocity()	85
10.27.1.10 Update()	85
10.27.1.11 UseRandomDirection()	85
10.27.1.12 UseRandomLifetime()	86
10.27.1.13 UseRandomVelocity()	86
10.28 CPlayer Class Reference	87
10.28.1 Member Function Documentation	87
10.28.1.1 Update()	87
10.29 CPlayerController Class Reference	87
10.29.1 Member Function Documentation	88
10.29.1.1 HandleInput()	88
10.29.1.2 OnPossess()	88
10.29.1.3 OnUnpossess()	88
10.30 CRigidBodyComponent Class Reference	89
10.30.1 Member Function Documentation	89
10.30.1.1 Draw()	89
10.30.1.2 GetAcceleration()	90
10.30.1.3 GetVelocity()	90
10.30.1.4 SetAcceleration()	90
10.30.1.5 SetVelocity()	90
10.30.1.6 Update()	91
10.31 Crossbow Class Reference	91
10.31.1 Member Function Documentation	91
10.31.1.1 Update()	92
10.32 CSpriteComponent Class Reference	93
10.32.1 Detailed Description	94
10.32.2 Member Function Documentation	94
10.32.2.1 Draw()	94
10.32.2.2 GetTransform()	94
10.32.2.3 LoadTexture()	94
10.32.2.4 LoadTextureWIC()	94
10.32.2.5 SetRenderRect()	95
10.32.2.6 SetSpriteSize()	95

10.32.2.7 SetTextureOffset()	95
10.32.2.8 SetUseTranslucency()	95
10.32.2.9 Update()	95
10.33 CT_EditorEntity Class Reference	96
10.33.1 Member Function Documentation	96
10.33.1.1 Update()	96
10.34 CT_EditorEntity_Enemy Class Reference	97
10.34.1 Member Function Documentation	98
10.34.1.1 InitialiseEntity()	98
10.34.1.2 Update()	98
10.35 CT_EditorEntity_PlayerStart Class Reference	98
10.35.1 Member Function Documentation	99
10.35.1.1 Update()	99
10.36 CT_EditorEntity_Waypoint Class Reference	99
10.36.1 Member Function Documentation	100
10.36.1.1 InitialiseEntity()	100
10.36.1.2 Update()	100
10.37 CT_EditorEntity_WeaponHolder Class Reference	100
10.37.1 Member Function Documentation	101
10.37.1.1 InitialiseEntity()	101
10.37.1.2 Update()	101
10.38 CT_EditorGrid Class Reference	102
10.38.1 Member Function Documentation	102
10.38.1.1 Update()	102
10.39 CT_EditorMain Class Reference	103
10.40 CT_EditorWindows Class Reference	103
10.41 CT_PropData Struct Reference	103
10.42 CTextRenderComponent Class Reference	104
10.42.1 Detailed Description	105
10.42.2 Member Function Documentation	105
10.42.2.1 Draw()	105
10.42.2.2 SetCharacterSize()	105
10.42.2.3 SetJustification()	105
10.42.2.4 SetReserveCount()	105
10.42.2.5 SetSpriteSheetColumnsCount()	106
10.42.2.6 Update()	106
10.43 CTexture Struct Reference	106
10.43.1 Detailed Description	106
10.44 CTile Class Reference	107
10.44.1 Member Function Documentation	107
10.44.1.1 Update()	108
10.45 CTransform Class Reference	108

10.45.1 Detailed Description	109
10.46 CUIManager Class Reference	109
10.47 CursorEntity Class Reference	110
10.47.1 Member Function Documentation	110
10.47.1.1 Update() [1/2]	110
10.47.1.2 Update() [2/2]	110
10.48 CWidget Class Reference	111
10.48.1 Member Function Documentation	111
10.48.1.1 AddChild()	111
10.48.1.2 SetVisibility()	112
10.48.1.3 SetWidgetTransform()	112
10.49 CWidget_Button Class Reference	112
10.49.1 Member Function Documentation	113
10.49.1.1 Bind_HoverEnd()	114
10.49.1.2 Bind_HoverStart()	114
10.49.1.3 Bind_OnButtonPressed()	114
10.49.1.4 Bind_OnButtonReleased()	114
10.49.1.5 SetButtonSize()	115
10.49.1.6 SetText()	115
10.49.1.7 SetTexture()	115
10.49.1.8 SetVisibility()	116
10.49.1.9 SetWidgetTransform()	116
10.49.1.10 Update()	116
10.50 CWidget_Canvas Class Reference	117
10.50.1 Member Function Documentation	117
10.50.1.1 GetMousePosition()	118
10.50.1.2 InitialiseCanvas()	118
10.50.1.3 SetVisibility()	118
10.50.1.4 Update()	118
10.51 CWidget_Image Class Reference	119
10.51.1 Member Function Documentation	119
10.51.1.1 SetVisibility()	119
10.51.1.2 SetWidgetTransform()	120
10.51.1.3 Update()	120
10.52 CWidget_Text Class Reference	121
10.52.1 Member Function Documentation	121
10.52.1.1 SetVisibility()	121
10.52.1.2 SetWidgetTransform()	122
10.52.1.3 Update()	122
10.53 CWorld Class Reference	122
10.53.1 Member Data Documentation	123
10.53.1.1 mapSize	123

10.54 CWorld_Editable Class Reference	124
10.54.1 Member Function Documentation	125
10.54.1.1 LoadWorld()	125
10.54.1.2 SetupWorld()	125
10.54.1.3 UnloadWorld()	125
10.55 CWorld_Game Class Reference	125
10.55.1 Constructor & Destructor Documentation	126
10.55.1.1 CWorld_Game()	126
10.55.2 Member Function Documentation	126
10.55.2.1 LoadEntities()	126
10.55.2.2 ReloadWorld()	126
10.55.2.3 SetupWorld()	127
10.55.2.4 UnloadWorld()	127
10.56 CWorld_Menu Class Reference	127
10.57 CWorldManager Class Reference	127
10.57.1 Member Function Documentation	128
10.57.1.1 LoadWorld() [1/3]	128
10.57.1.2 LoadWorld() [2/3]	128
10.57.1.3 LoadWorld() [3/3]	128
10.58 Dagger Class Reference	129
10.59 Debug Class Reference	129
10.59.1 Member Function Documentation	130
10.59.1.1 GetLogging()	130
10.59.1.2 getOutput()	130
10.59.1.3 GetVisibility()	130
10.59.1.4 Log()	130
10.59.1.5 LogError()	131
10.59.1.6 LogHResult()	131
10.59.1.7 SetLogging()	131
10.59.1.8 SetVisibility()	132
10.60 DebugOutput Class Reference	132
10.61 Dialogue Struct Reference	132
10.62 DialogueHandler Class Reference	133
10.62.1 Member Function Documentation	133
10.62.1.1 AdvanceDialogue()	133
10.62.1.2 LoadDialogue()	134
10.62.1.3 SetDialogue()	134
10.63 DialogueUI Class Reference	134
10.63.1 Detailed Description	135
10.63.2 Member Function Documentation	135
10.63.2.1 Advance()	135
10.63.2.2 SetName()	135

10.63.2.3 SetText()	136
10.63.2.4 ToggleDrawing()	136
10.63.2.5 Update()	136
10.64 DogEnemy Class Reference	137
10.64.1 Detailed Description	137
10.64.2 Member Function Documentation	137
10.64.2.1 AttackEnter()	137
10.64.2.2 AttackPlayer()	138
10.64.2.3 ChasePlayer()	138
10.64.2.4 OnDeath()	138
10.64.2.5 OnHit()	139
10.64.2.6 Update()	139
10.65 Engine Struct Reference	139
10.66 EntityManager Class Reference	140
10.66.1 Detailed Description	140
10.66.2 Member Function Documentation	140
10.66.2.1 RemoveComponent()	140
10.66.2.2 RemoveEntity()	141
10.66.2.3 SortTranslucentComponents()	141
10.67 EventSystem Class Reference	141
10.67.1 Member Function Documentation	141
10.67.1.1 AddListener()	141
10.67.1.2 TriggerEvent()	142
10.68 Fireball Class Reference	142
10.69 GruntEnemy Class Reference	143
10.69.1 Detailed Description	143
10.69.2 Member Function Documentation	143
10.69.2.1 AttackPlayer()	143
10.69.2.2 ChasePlayer()	144
10.69.2.3 OnDeath()	144
10.69.2.4 OnHit()	144
10.69.2.5 Update()	144
10.70 HomingProjectile Class Reference	145
10.70.1 Member Function Documentation	145
10.70.1.1 Update()	145
10.71 Inputable Class Reference	146
10.71.1 Member Function Documentation	146
10.71.1.1 PressedDrop()	146
10.71.1.2 PressedHorizontal()	146
10.71.1.3 PressedInteract()	146
10.71.1.4 PressedVertical()	147
10.72 InputManager Class Reference	147

10.73 InvestigateState Class Reference	148
10.73.1 Detailed Description	148
10.73.2 Member Function Documentation	148
10.73.2.1 Enter()	149
10.73.2.2 Exit()	149
10.73.2.3 Update()	149
10.74 InvisibilityScroll Class Reference	149
10.75 IO Class Reference	150
10.75.1 Member Function Documentation	150
10.75.1.1 FindExtension()	150
10.76 IUsePickup Class Reference	150
10.76.1 Member Function Documentation	151
10.76.1.1 UsePickup()	151
10.77 LevelCompleteMenu Class Reference	151
10.78 LevelSelectMenu Class Reference	152
10.79 LevelTransporter Class Reference	153
10.79.1 Member Function Documentation	153
10.79.1.1 OnInteract()	153
10.80 Longsword Class Reference	154
10.80.1 Member Function Documentation	154
10.80.1.1 OnFire()	154
10.81 MagicMissile Class Reference	155
10.81.1 Member Function Documentation	155
10.81.1.1 OnFire()	155
10.82 MainMenu Class Reference	156
10.83 MaterialPropertiesConstantBuffer Struct Reference	156
10.84 Math Class Reference	156
10.84.1 Detailed Description	157
10.84.2 Member Function Documentation	157
10.84.2.1 FloatToStringWithDigits()	157
10.84.2.2 FromScreenToWorld()	158
10.84.2.3 IntToString()	158
10.85 MeleeWeapon Class Reference	158
10.85.1 Member Function Documentation	159
10.85.1.1 OnFire()	159
10.86 NecrodoggieconPage Class Reference	160
10.86.1 Member Function Documentation	160
10.86.1.1 OnInteract()	160
10.87 Pathfinding Class Reference	161
10.87.1 Detailed Description	161
10.87.2 Constructor & Destructor Documentation	161
10.87.2.1 Pathfinding()	161

10.87.3 Member Function Documentation	162
10.87.3.1 CalculateCost()	162
10.87.3.2 CalculatePath()	162
10.87.3.3 FindClosestPatrolNode()	162
10.87.3.4 FindClosestWaypoint()	164
10.87.3.5 GetPathNodes()	164
10.87.3.6 SetPath()	164
10.87.3.7 SetPatrolNodes()	165
10.88 PatrolNode Struct Reference	165
10.88.1 Detailed Description	165
10.89 PatrolState Class Reference	166
10.89.1 Detailed Description	166
10.89.2 Member Function Documentation	166
10.89.2.1 Enter()	166
10.89.2.2 Exit()	166
10.89.2.3 Update()	167
10.90 PauseMenu Class Reference	167
10.90.1 Member Function Documentation	167
10.90.1.1 Update()	168
10.91 Pickup Class Reference	168
10.91.1 Member Function Documentation	168
10.91.1.1 OnFire()	169
10.91.1.2 Update()	170
10.92 PlayerCharacter Class Reference	170
10.92.1 Member Function Documentation	172
10.92.1.1 ApplyDamage() [1/2]	172
10.92.1.2 ApplyDamage() [2/2]	172
10.92.1.3 Attack()	172
10.92.1.4 PressedDrop()	173
10.92.1.5 PressedHorizontal()	173
10.92.1.6 PressedInteract()	173
10.92.1.7 PressedUse()	173
10.92.1.8 PressedVertical()	173
10.92.1.9 ToggleVisibility()	174
10.92.1.10 Update()	174
10.92.1.11 UsePickup()	174
10.93 PlayerController Class Reference	175
10.93.1 Member Function Documentation	175
10.93.1.1 HandleInput()	175
10.93.1.2 OnPossess()	176
10.93.1.3 OnUnpossess()	176
10.93.1.4 Update()	176

10.94 Projectile Class Reference	177
10.94.1 Detailed Description	177
10.94.2 Member Function Documentation	177
10.94.2.1 DidItHit()	178
10.94.2.2 StartUp()	178
10.94.2.3 Update()	178
10.95 PropData Struct Reference	178
10.96 RangeWeapon Class Reference	179
10.96.1 Member Function Documentation	179
10.96.1.1 OnFire()	179
10.97 Rapier Class Reference	180
10.98 SearchState Class Reference	180
10.98.1 Detailed Description	181
10.98.2 Member Function Documentation	181
10.98.2.1 Enter()	181
10.98.2.2 Exit()	181
10.98.2.3 Update()	181
10.99 SettingsMenu Class Reference	182
10.99.1 Member Function Documentation	182
10.99.1.1 Update()	182
10.100 ShieldScroll Class Reference	183
10.101 SimpleVertex Struct Reference	183
10.102 SoundManager Class Reference	183
10.102.1 Member Function Documentation	184
10.102.1.1 AddSound() [1/2]	184
10.102.1.2 AddSound() [2/2]	184
10.102.1.3 PlayMusic()	185
10.102.1.4 PlaySound()	185
10.103 State Class Reference	185
10.103.1 Detailed Description	186
10.104 TestUI Class Reference	186
10.104.1 Member Function Documentation	186
10.104.1.1 Update()	187
10.105 Vector2Base< T > Class Template Reference	187
10.106 Vector3Base< T > Class Template Reference	188
10.107 WaypointNode Struct Reference	189
10.107.1 Detailed Description	189
10.108 Weapon Class Reference	190
10.108.1 Detailed Description	191
10.108.2 Member Function Documentation	191
10.108.2.1 Draw()	191
10.108.2.2 OnFire()	191

10.108.2.3 SetWeapon()	192
10.108.2.4 Update()	192
10.109 WeaponInterface Class Reference	192
10.109.1 Detailed Description	193
10.109.2 Member Function Documentation	193
10.109.2.1 Draw()	193
10.109.2.2 OnFire()	194
10.109.2.3 SetUserType()	194
10.109.2.4 SetWeapon()	194
10.109.2.5 Update()	194
10.110 WeaponPickup< T > Class Template Reference	195
10.110.1 Member Function Documentation	195
10.110.1.1 OnInteract()	195
10.110.1.2 SetWeapon()	195
10.111 weaponUI Class Reference	196
10.111.1 Member Function Documentation	196
10.111.1.1 Update()	196
10.111.1.2 updateUI()	197
11 File Documentation	199
11.1 CAINode.h File Reference	199
11.1.1 Detailed Description	199
11.2 CAINode.h	200
11.3 Pathfinding.cpp File Reference	200
11.3.1 Detailed Description	200
11.4 Pathfinding.h File Reference	200
11.4.1 Detailed Description	201
11.5 Pathfinding.h	201
11.6 CComponent.h File Reference	201
11.6.1 Detailed Description	202
11.7 CComponent.h	202
11.8 CEntity.h File Reference	203
11.8.1 Detailed Description	203
11.9 CEntity.h	203
11.10 CAnimationSpriteComponent.h File Reference	204
11.10.1 Detailed Description	205
11.11 CAnimationSpriteComponent.h	205
11.12 CAudioEmitterComponent.h File Reference	205
11.12.1 Detailed Description	206
11.13 CAudioEmitterComponent.h	206
11.14 CCameraComponent.h File Reference	206
11.14.1 Detailed Description	207

11.15 CCameraComponent.h	207
11.16 CParticleEmitter.h File Reference	208
11.16.1 Detailed Description	208
11.17 CParticleEmitter.h	208
11.18 CRigidBodyComponent.cpp File Reference	209
11.18.1 Detailed Description	209
11.19 CRigidBodyComponent.h	210
11.20 CSpriteComponent.h File Reference	210
11.20.1 Detailed Description	210
11.21 CSpriteComponent.h	211
11.22 CTextRenderComponent.h File Reference	211
11.22.1 Detailed Description	212
11.22.2 Enumeration Type Documentation	212
11.22.2.1 TextJustification	212
11.23 CTextRenderComponent.h	212
11.24 Engine.h	213
11.25 CParticle.cpp File Reference	214
11.25.1 Detailed Description	214
11.26 CParticle.h	214
11.27 CGridCursor.h	215
11.28 CTile.h	215
11.29 CWorld.h	216
11.30 CWorld_Edit.h	218
11.31 IInputable.h	221
11.32 CCamera.h File Reference	221
11.32.1 Detailed Description	221
11.33 CCamera.h	221
11.34 CMaterial.h File Reference	222
11.34.1 Detailed Description	222
11.35 CMaterial.h	222
11.36 CMesh.h File Reference	223
11.36.1 Detailed Description	223
11.37 CMesh.h	223
11.38 CTexture.h File Reference	224
11.38.1 Detailed Description	224
11.39 CTexture.h	224
11.40 structures.h	224
11.41 CWidget.cpp File Reference	225
11.41.1 Detailed Description	225
11.42 CWidget.h	225
11.43 CWidget_Button.cpp File Reference	225
11.43.1 Detailed Description	226

11.44 CWidget_Button.h	226
11.45 CWidget_Canvas.h File Reference	227
11.45.1 Detailed Description	227
11.46 CWidget_Canvas.h	228
11.47 CWidget_Image.h File Reference	228
11.47.1 Detailed Description	228
11.48 CWidget_Image.h	229
11.49 CWidget_Text.h	229
11.50 AssetManager.h File Reference	229
11.50.1 Detailed Description	230
11.51 AssetManager.h	230
11.52 AudioController.h File Reference	230
11.52.1 Detailed Description	231
11.53 AudioController.h	231
11.54 CAudio.h File Reference	232
11.54.1 Detailed Description	232
11.55 CAudio.h	232
11.56 CEmitter.h File Reference	232
11.56.1 Detailed Description	233
11.57 CEmitter.h	233
11.58 CameraManager.h File Reference	233
11.58.1 Detailed Description	234
11.59 CameraManager.h	234
11.60 CollisionComponent.h	234
11.61 CTransform.h File Reference	235
11.61.1 Detailed Description	235
11.62 CTransform.h	236
11.63 CUIManager.h	236
11.64 CWorldManager.h	236
11.65 Debug.h File Reference	237
11.65.1 Detailed Description	237
11.66 Debug.h	238
11.67 DebugOutput.h	240
11.68 EntityManager.h File Reference	241
11.68.1 Detailed Description	242
11.69 EntityManager.h	242
11.70 EventSystem.h File Reference	242
11.70.1 Detailed Description	243
11.71 EventSystem.h	243
11.72 InputManager.cpp File Reference	243
11.72.1 Detailed Description	243
11.73 InputManager.h File Reference	244

11.73.1 Detailed Description	244
11.74 InputManager.h	244
11.75 IO.h File Reference	246
11.75.1 Detailed Description	246
11.76 IO.h	246
11.77 Math.h File Reference	247
11.77.1 Detailed Description	247
11.78 Math.h	247
11.79 Vector3.h	248
11.80 Cerberus/Resource.h	252
11.81 Necrodoggiecon/Resource.h	252
11.82 CT_EditorEntity.h	253
11.83 CT_EditorGrid.h	255
11.84 CT_EditorMain.h	256
11.85 CT_EditorWindows.h	256
11.86 WorldConstants.h	257
11.87 CerberusTools/CursorEntity.h	258
11.88 Necrodoggiecon/Game/CursorEntity.h	258
11.89 CWorld_Game.h	259
11.90 CWorld_Menu.h	259
11.91 AlarmEnemy.cpp File Reference	259
11.91.1 Detailed Description	259
11.92 AlarmEnemy.h File Reference	260
11.92.1 Detailed Description	260
11.93 AlarmEnemy.h	260
11.94 CAIController.cpp File Reference	260
11.94.1 Detailed Description	261
11.95 CAIController.h File Reference	261
11.95.1 Detailed Description	261
11.96 CAIController.h	262
11.97 DogEnemy.cpp File Reference	263
11.97.1 Detailed Description	264
11.98 DogEnemy.h File Reference	264
11.98.1 Detailed Description	264
11.99 DogEnemy.h	264
11.100 GruntEnemy.cpp File Reference	265
11.100.1 Detailed Description	265
11.101 GruntEnemy.h File Reference	265
11.101.1 Detailed Description	265
11.102 GruntEnemy.h	266
11.103 State.cpp File Reference	266
11.103.1 Detailed Description	266

11.104 State.h File Reference	266
11.104.1 Detailed Description	267
11.105 State.h	267
11.106 AudioEmitterEntity.cpp File Reference	268
11.106.1 Detailed Description	268
11.107 AudioEmitterEntity.h	269
11.108 CCharacter.cpp File Reference	269
11.108.1 Detailed Description	269
11.109 CCharacter.h	269
11.110 CInteractable.h File Reference	270
11.110.1 Detailed Description	270
11.111 CInteractable.h	271
11.112 CPlayer.h	271
11.113 CPlayerController.cpp File Reference	271
11.113.1 Detailed Description	272
11.114 CPlayerController.h	272
11.115 Dialogue.h	272
11.116 DialogueHandler.cpp File Reference	273
11.116.1 Detailed Description	273
11.117 DialogueHandler.h	273
11.118 DialogueUI.cpp File Reference	273
11.118.1 Detailed Description	274
11.119 DialogueUI.h	274
11.120 IUsePickup.h	274
11.121 LevelTransporter.h	275
11.122 NecrodoggieconPage.h	275
11.123 PlayerCharacter.h	275
11.124 PlayerController.h	276
11.125 SoundManager.cpp File Reference	276
11.125.1 Detailed Description	277
11.126 SoundManager.h	277
11.127 TestUI.h	277
11.128 WeaponInterface.h File Reference	278
11.128.1 Detailed Description	278
11.129 WeaponInterface.h	278
11.130 WeaponPickup.h File Reference	279
11.130.1 Detailed Description	279
11.131 WeaponPickup.h	279
11.132 weapons.h File Reference	281
11.132.1 Detailed Description	281
11.133 weapons.h	282
11.134 HomingProjectile.cpp File Reference	283

11.134.1 Detailed Description	283
11.135 HomingProjectile.h File Reference	283
11.135.1 Detailed Description	284
11.136 HomingProjectile.h	284
11.137 LevelCompleteMenu.cpp File Reference	284
11.137.1 Detailed Description	284
11.138 LevelCompleteMenu.h File Reference	285
11.138.1 Detailed Description	285
11.139 LevelCompleteMenu.h	285
11.140 LevelSelectMenu.cpp File Reference	285
11.140.1 Detailed Description	286
11.141 LevelSelectMenu.h File Reference	286
11.141.1 Detailed Description	286
11.142 LevelSelectMenu.h	287
11.143 MainMenu.cpp File Reference	287
11.143.1 Detailed Description	287
11.144 MainMenu.h File Reference	288
11.144.1 Detailed Description	288
11.145 MainMenu.h	288
11.146 PauseMenu.cpp File Reference	289
11.146.1 Detailed Description	289
11.147 PauseMenu.h File Reference	289
11.147.1 Detailed Description	289
11.148 PauseMenu.h	290
11.149 Projectile.cpp File Reference	290
11.149.1 Detailed Description	290
11.150 Projectile.h File Reference	290
11.150.1 Detailed Description	291
11.151 Projectile.h	291
11.152 SettingsMenu.cpp File Reference	292
11.152.1 Detailed Description	292
11.153 SettingsMenu.h File Reference	292
11.153.1 Detailed Description	293
11.154 SettingsMenu.h	293
11.155 Dagger.h File Reference	293
11.155.1 Detailed Description	294
11.156 Dagger.h	294
11.157 Longsword.h File Reference	294
11.157.1 Detailed Description	294
11.158 Longsword.h	295
11.159 Rapier.h File Reference	295
11.159.1 Detailed Description	295

11.160 Rapier.h	295
11.161 MeleeWeapon.cpp File Reference	296
11.161.1 Detailed Description	296
11.162 MeleeWeapon.h	296
11.163 Pickup.cpp File Reference	296
11.163.1 Detailed Description	297
11.164 Pickup.h	297
11.165 InvisibilityScroll.h	297
11.166 ShieldScroll.h	297
11.167 Crossbow.cpp File Reference	297
11.167.1 Detailed Description	298
11.168 Crossbow.h File Reference	298
11.168.1 Detailed Description	298
11.169 Crossbow.h	298
11.170 Fireball.cpp File Reference	299
11.170.1 Detailed Description	299
11.171 Fireball.h File Reference	299
11.171.1 Detailed Description	299
11.172 Fireball.h	300
11.173 MagicMissile.cpp File Reference	300
11.173.1 Detailed Description	300
11.174 MagicMissile.h File Reference	300
11.174.1 Detailed Description	301
11.175 MagicMissile.h	301
11.176 RangeWeapon.h	301
11.177 weaponUI.cpp File Reference	301
11.177.1 Detailed Description	302
11.178 weaponUI.h File Reference	302
11.178.1 Detailed Description	302
11.179 weaponUI.h	302
Index	303

Chapter 1

Necrodoggiecon

1.1 How the project works

The engine holds all the intrinsic components and the other outer projects can create classes that inherit these components and then the class can be used to create the game on top of the engine.

1.2 Instructions

1.2.1 How to compile the Engine

Open the CerberusEngine project in Visual Studio. Right click on the project and click build. Do not debug the engine, just build it.

1.2.2 How to compile the Game

Firstly, compile the engine. Open the Necrodoggiecon project in Visual Studio. Right click on the project and set as startup project. Set to release mode and press F5 to run the game.

1.2.3 How to play the Game

1.2.3.1 Controls:

- WASD - Movement
- Left click - Fire weapon
- F - Interact

1.2.4 Naming Convention

1.2.4.1 Variables:

varNameHere.

1.2.4.2 Functions:

FunctionNameHere.

1.2.4.3 Enums, Defines:

ANGRYENUMS

1.2.5 Links to the aspects of the Engine/Game

- [AI](#)
- [AssetManagement](#)
- [Audio](#)
- [Utility](#)
- [Weapons](#)

Chapter 2

AI

2.1 Navigation and Pathfinding

The AI uses the A* algorithm with waypoints to navigate through the level and this is handled in the [Pathfinding](#) class. Firstly, all the walkable tiles are taken in a waypoints and converted into waypoint nodes. When the `SetPath` function is called, the start and end waypoint node is passed in and a vector of nodes is produced with the necessary waypoint nodes to traverse for the path.

2.2 Perception

The AI perception is done using a `CanSee` and `CanHear` function. The `CanSee` function checks to see if the player position is within the vision range and return true if that is the case. The `CanHear` function is a lambda function that is called whenever the `SoundPlayed` event occurs. It gets all the emitters in range that are playing and return the position of the closest one. The AI will then investigate this position.

2.3 Decision Making using a Finite State Machine

The AI uses a Finite [State](#) Machine detailed in the [State](#) Class. The FSM is implemented using a base state class and the different states are inherited. The different states are:

- [PatrolState](#)
- [ChaseState](#)
- [AttackState](#)
- [InvestigateState](#)
- [SearchState](#)

These are setup so that the state machine can be built for any enemy but can also call the specific functions for each state. Each state has a enter, update and exit function. The enter and exit functions are called once on first switching to the state and upon switching out of the state. These functions are used for switching sprite to the relevant look for the state and to setting a path before traversing this path in the patrol state. This is the AI State Machine Diagram.

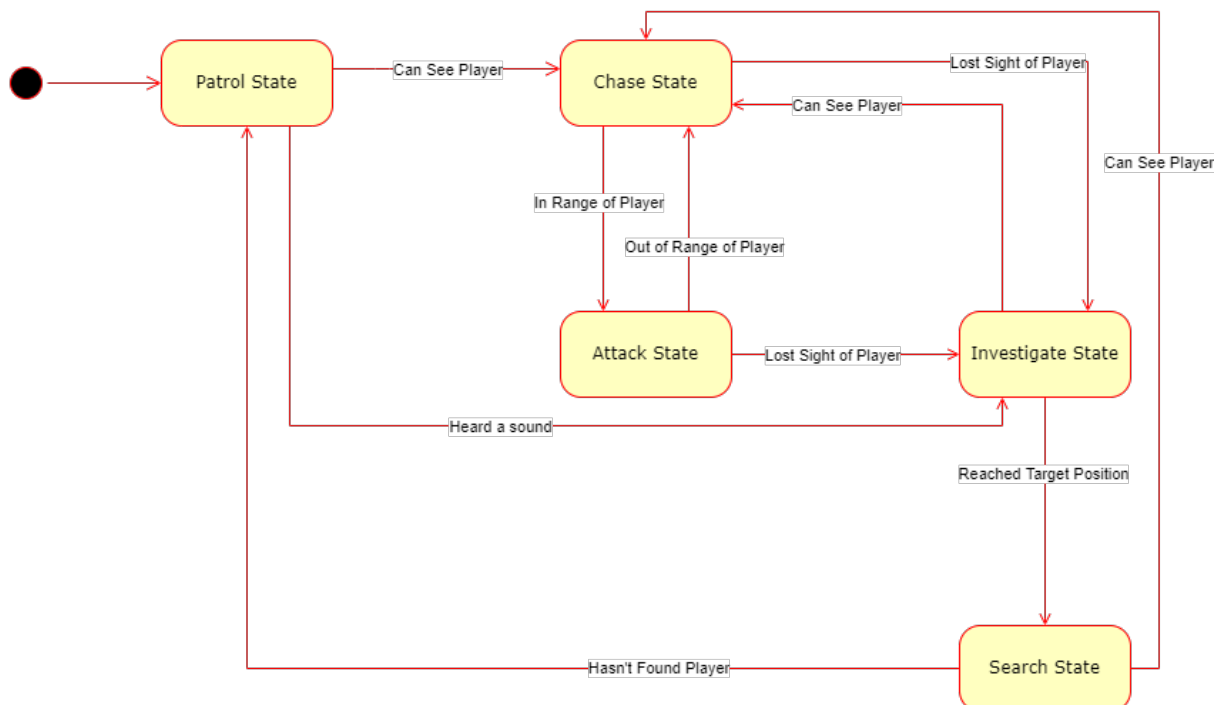


Figure 2.1 The Diagram

2.4 Enemies

All the enemies inherit from the [CAIController](#) class which acts as the base class for the AI behaviour. This class handles the movement of the enemies and the view semi circle. It also handles the interaction with the pathfinding class the holds virtual functions to be overridden in the inherited classes. There are 3 types of enemies that inherited from [CAIController](#):

- [AlarmEnemy](#)
- [DogEnemy](#)
- [GruntEnemy](#)

2.4.1 AlarmEnemy

The [AlarmEnemy](#) is an enemy that will alert nearby enemies by playing the bell it is holding to make a sound if it sees the player. The nearby enemies will then head to the location of the bell sound. This enemy does not attack the player and is meant to punish the player if they are caught by it.

2.4.2 DogEnemy

The [DogEnemy](#) will attack the player using a dash. This works by using 2 timers, one for the attack and one for the cooldown. The dash is emulated by increasing the speed of the dog for a small period of time and the dog will dash in a straight line towards the player. The indication of when the dog is about to attack is done by slowing the dog drastically for a brief period of time before dashing.

2.4.3 GruntEnemy

The [GruntEnemy](#) holds a weapon and it will use the weapon if it gets within the weapon's range of the player. This works for both melee and ranged weapons.

2.5 Relating Classes:

- [CAIController](#)
- [Pathfinding](#)
- [State](#)
- [GruntEnemy](#)
- [AlarmEnemy](#)
- [DogEnemy](#)

Chapter 3

Asset Manager

3.1 Managing Assets within the engine.

The asset manager was created to allow for many sprites to be drawn to the screen without a enourmous overhead. This was done by only allocating memory once for a specific object. This means that all sprites in the scene can use the same mesh data and the engine doesnt have to re-generate the mesh data everytime a new object wishes to be spawned. Instead the engine polls the asset manager and the manager retrievees the data and passes it onto the caller. The caller can then instanciate objects with the data from the asset manager and skip the overhead of making stack memory itself. Furthermore, this has been extended for Audio and Textures to allow for those assets to be polled in a similar way.

3.2 Relating Classes:

- [AssetManager](#)

Chapter 4

Audio

4.1 Adding, Playing and Managing Audio

The audio system manages all audio in the game and abstracts away FMOD's lower level API. The audio manager also interfaces with the [AssetManager](#) to make sure that duplicate audio doesn't create unnecessary memory when not required. The audio manager is used heavily by audio emitters as a high-level abstraction to the audio system and FMOD. Furthermore, there are smaller classes used to store Audio data in a OOP way. For instance [CAudio](#) encapsulates all FMOD data into a easy to remove / change class, [CEmitter](#) operates in the same way but holds a [CAudio](#) reference and the range + other features of the emitter.

4.2 Audio Emitters

Audio Emitters are a component within the engine. This component is responsible for interfacing with the audio system to play audio at a certain location. The audio system keeps track of all emitters within the scene and attenuates them accordingly to allow for psuedo-3D audio.

4.3 Architecture

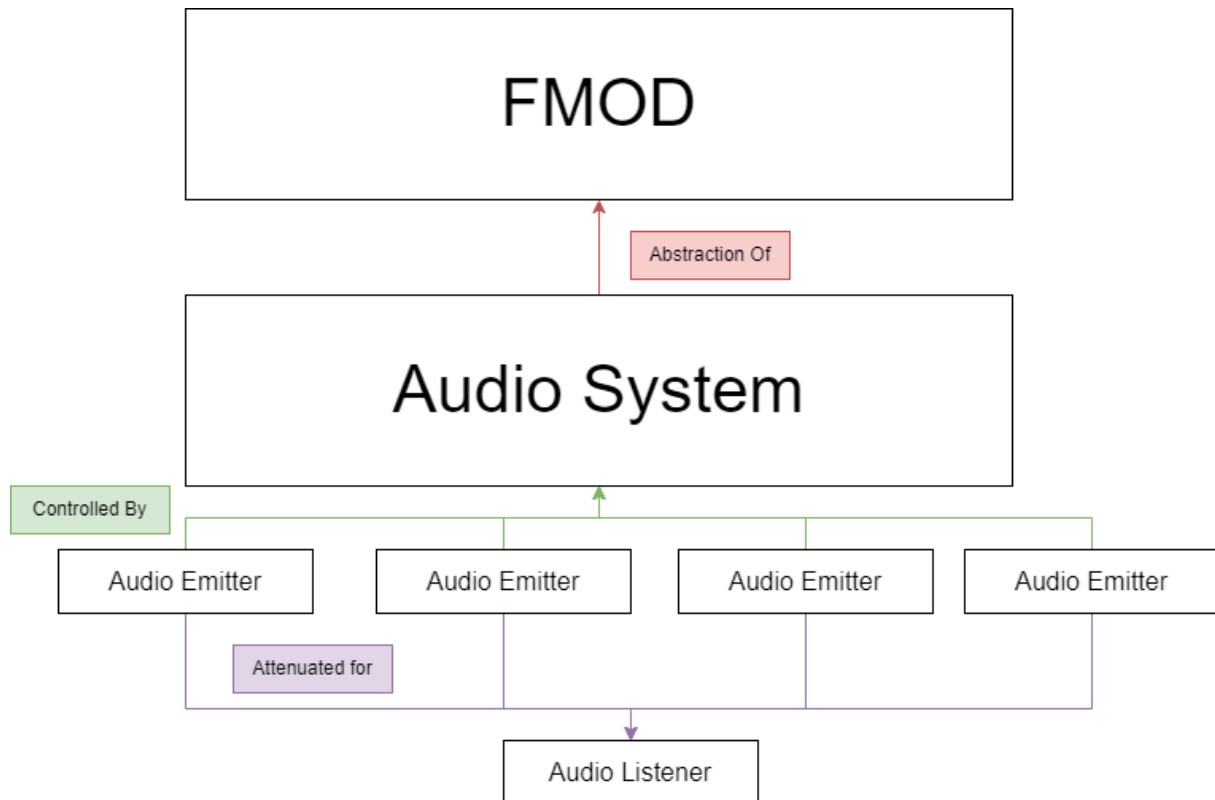


Figure 4.1 The General Architecture

4.4 Relating Classes:

- [AudioController](#)
- [CAudio](#)
- [CEmitter](#)
- [CAudioEmitterComponent](#)

Chapter 5

Utility

5.1 Relating Classes:

- [Vector2Base](#)
- [Vector3Base](#)
- [AssetManager](#)
- [AudioController](#)
- [CameraManager](#)
- [CollisionComponent](#)
- [Debug](#)
- [EventSystem](#)
- [InputManager](#)
- [Math](#)
- [CTransform](#)
- [CUIManager](#)
- [CWorldManager](#)
- [EntityManager](#)
- [IO](#)

Chapter 6

Weapon System

6.1 Strategy Design Pattern

The weapons system uses the Strategy Design Pattern to have a context interface that allows for multiple different strategies to be interchanged with their own unique logic. Firstly, the entities in the game (players and enemies) are given an instance of the context interface, this interface holds an instance of the base strategy. When a player or enemy changes their weapon, the strategy instance changes the pointer to the strategy it is using, for example, from [Dagger](#) to [Crossbow](#). This Design Pattern is great for a weapon system as it allows for weapons to be interchanged easily by only passing in the weapon pointer of the specific weapon and not having to have multiple objects created in memory for all weapons in the game.

6.2 Why does this Design Pattern work

The Strategy Design Pattern works because all the strategies that are being interchanged all inherit from the same base class ([Weapon](#) in this case). This means that all the subclasses of weapons that derive from the [Weapon](#) class and are strategies in the design pattern are all [Weapon](#) pointers at their base level because they have all inherited from [Weapon](#) at the base level of their inheritance tree. Each strategy in the design pattern is its own subclass, all with its own unique logic that is used depending on the strategy that is currently being used in the context interface. This is a basic relationship diagram of the weapon system implemented in the game.

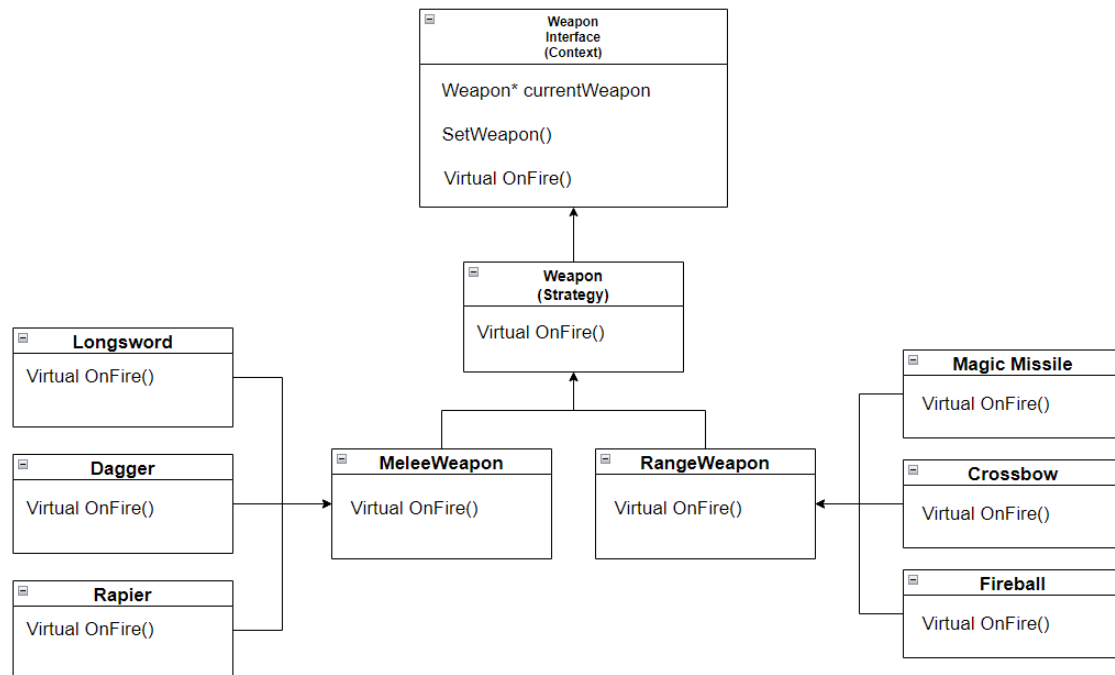


Figure 6.1 The Diagram

6.3 Weapons

The game has multiple different attack styles, Melee and Range weapons. All Melee and Range weapons have base variants which the subclasses inherit from. These base classes have the shared logic through all the weapons of its kind, an example of this is the basic projectile spawning for ranged weapons. This is then overridden through the unique logic in the individual weapon.

6.3.1 Melee Weapons

There are 3 melee weapons in the game:

- [Dagger](#)
- [Rapier](#)
- [Longsword](#)

The melee weapons in the game use a system of calculating the damage position based on the direction of the attack and the range of the weapon. This damage position is then used to get the enemy that is in range of the player and the damage position, and then discarding the all enemies until the closest enemy in range is returned. All the melee weapons use this method, dynamically calculating the damage position based on the different ranges of the weapons.

The [Longsword](#) has unique logic that creates an area-of-effect (AOE) attack using the same method that is used for the other melee weapons. However, all entities that are within the range of weapons from the player AND within the range of the weapon from the damage position are damaged. This radius style range from 2 points creates a cone shape in the looking direction.

6.3.2 Range Weapons

There are 3 range weapons in the game:

- [Crossbow](#)
- [Fireball](#)
- Magic Missile (Homing)

The range weapons in the game use a [Projectile](#) class to spawn a [CEntity](#) into the world with a given direction, speed, position and sprite. These parameters are then used to constantly update the entity on a constant velocity. The projectile also uses the same method of checking for closest entity in a given range around the projectile, this makes a sort of bounding area and if any entity is returned, then damage logic is applied to said entity.

The Magic Missile has unique logic that creates a Homing [Projectile](#) entity into the world. This projectile creates a directional vector to the closest entity it finds in a given range, and then travels along that new direction vector towards the target.

6.4 Relating Classes:

- [WeaponInterface](#)
- [Weapon](#)
- [MeleeWeapon](#)
- [RangeWeapon](#)
- [Dagger](#)
- [Rapier](#)
- [Longsword](#)
- [Crossbow](#)
- [MagicMissile](#)
- [Fireball](#)
- [Projectile](#)
- [HomingProjectile](#)

Chapter 7

Hierarchical Index

7.1 Class Hierarchy

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

_Material	31
AssetManager	33
AudioController	38
CameraManager	52
CAudio	56
CellData	67
CEmitter	68
CMaterial	76
CMesh	76
CollisionComponent	77
ConstantBuffer	78
CT_EditorMain	103
CT_EditorWindows	103
CT_PropData	103
CTexture	106
CTransform	108
CComponent	65
CAudioEmitterComponent	56
CCameraComponent	60
CParticleEmitter	81
CRigidBodyComponent	89
CSpriteComponent	93
CAnimationSpriteComponent	54
CTextRenderComponent	104
Weapon	190
MeleeWeapon	158
Dagger	129
Longsword	154
Rapier	180
Pickup	168
InvisibilityScroll	149
ShieldScroll	183
RangeWeapon	179
Crossbow	91
Fireball	142

MagicMissile	155
WeaponInterface	192
CEntity	68
AudioEmitterEntity	42
CCamera	59
CCharacter	63
CAIController	45
AlarmEnemy	31
DogEnemy	137
GruntEnemy	143
PlayerCharacter	170
CGridCursor	70
CInteractable	73
LevelTransporter	153
NecrodoggieconPage	160
WeaponPickup< T >	195
CParticle	78
CPlayer	87
CPlayerController	87
PlayerController	175
CT_EditorEntity	96
CT_EditorEntity_Enemy	97
CT_EditorEntity_PlayerStart	98
CT_EditorEntity_Waypoint	99
CT_EditorEntity_WeaponHolder	100
CT_EditorGrid	102
CTile	107
CWidget	111
CWidget_Button	112
CWidget_Canvas	117
LevelCompleteMenu	151
LevelSelectMenu	152
MainMenu	156
PauseMenu	167
SettingsMenu	182
CWidget_Image	119
CWidget_Text	121
CursorEntity	110
CursorEntity	110
DialogueHandler	133
DialogueUI	134
Projectile	177
HomingProjectile	145
SoundManager	183
TestUI	186
weaponUI	196
CUIManager	109
CWorld	122
CWorld_Editable	124
CWorld_Game	125
CWorld_Menu	127
CWorldManager	127
Debug	129
DebugOutput	132
Dialogue	132
Engine	139
EntityManager	140

EventSystem	141
IInputable	146
PlayerCharacter	170
InputManager	147
IO	150
IUsePickup	150
PlayerCharacter	170
MaterialPropertiesConstantBuffer	156
Math	156
Pathfinding	161
PatrolNode	165
PropData	178
SimpleVertex	183
State	185
AttackState	36
ChaseState	71
InvestigateState	148
PatrolState	166
SearchState	180
Vector2Base< T >	187
Vector2Base< float >	187
Vector3Base< T >	188
Vector3Base< float >	188
WaypointNode	189

Chapter 8

Class Index

8.1 Class List

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

_Material	31
AlarmEnemy	
Class for the alarm enemy	31
AssetManager	33
AttackState	
State for when the AI is attacking the player	36
AudioController	38
AudioEmitterEntity	42
CAIController	
Controller class for the AI	45
CameraManager	52
CAnimationSpriteComponent	
Extends CSpriteComponent to automatically animate sprite-sheets	54
CAudio	56
CAudioEmitterComponent	56
CCamera	59
CCameraComponent	60
CCharacter	63
CComponent	
Fundamental component class of the engine	65
CellData	67
CEmitter	68
CEntity	
Fundamental class of the engine with a world transform and ability to have components	68
CGridCursor	70
ChaseState	
State for when the AI is chasing the player	71
CInteractable	73
CMaterial	
Holds the directx stuff for uploading sprite specific data to the shader	76
CMesh	
Holds all information about a mesh for use by CSpriteComponent	76
CollisionComponent	77
ConstantBuffer	78
CParticle	78

CParticleEmitter	81
CPlayer	87
CPlayerController	87
CRigidBodyComponent	89
Crossbow	91
CSpriteComponent	
A component for loading and displaying a 2D texture in world space as part of CEntity	93
CT_EditorEntity	96
CT_EditorEntity_Enemy	97
CT_EditorEntity_PlayerStart	98
CT_EditorEntity_Waypoint	99
CT_EditorEntity_WeaponHolder	100
CT_EditorGrid	102
CT_EditorMain	103
CT_EditorWindows	103
CT_PropData	103
CTextRenderComponent	
A component for rendering text to the screen from a sprite-sheet	104
CTexture	
Holds all information about a texture for use by CSpriteComponent	106
CTile	107
CTransform	
A transform class that contains getters and setters	108
CUIManager	109
CursorEntity	110
CWidget	111
CWidget_Button	112
CWidget_Canvas	117
CWidget_Image	119
CWidget_Text	121
CWorld	122
CWorld_Editable	124
CWorld_Game	125
CWorld_Menu	127
CWorldManager	127
Dagger	129
Debug	129
DebugOutput	132
Dialogue	132
DialogueHandler	133
DialogueUI	
Class that handles displaying text in the dialogue window	134
DogEnemy	
Class for the dog enemy	137
Engine	139
EntityManager	
Static class for tracking entities and components while accommodating translucency	140
EventSystem	141
Fireball	142
GruntEnemy	
Class for the Grunt enemy	143
HomingProjectile	145
IInputable	146
InputManager	147
InvestigateState	
State for when the AI is investigating	148
InvisibilityScroll	149
IO	150

IUsePickup	150
LevelCompleteMenu	151
LevelSelectMenu	152
LevelTransporter	153
Longsword	154
MagicMissile	155
MainMenu	156
MaterialPropertiesConstantBuffer	156
Math	
Class of all the static maths functions that don't fit into existing classes	156
MeleeWeapon	158
NecrodoggieconPage	160
Pathfinding	
Pathfinding class to handle all the pathfinding for the AI	161
PatrolNode	
Patrol node struct containing the position, closest waypoint and the next patrol node	165
PatrolState	
State for when the AI is patrolling between the patrol points	166
PauseMenu	167
Pickup	168
PlayerCharacter	170
PlayerController	175
Projectile	
Projectile class for the Projectile	177
PropData	178
RangeWeapon	179
Rapier	180
SearchState	
State for when the AI is searching for the player	180
SettingsMenu	182
ShieldScroll	183
SimpleVertex	183
SoundManager	183
State	
Base state class	185
TestUI	186
Vector2Base< T >	187
Vector3Base< T >	188
WaypointNode	
Waypoint node struct containing the waypoint, parent waypoint, neighbours and the costs	189
Weapon	
Base Weapon class inherited by all weapons	190
WeaponInterface	
Weapon Interface class used to switch weapons being used through the Strategy Design Pattern	192
WeaponPickup< T >	195
weaponUI	196

Chapter 9

File Index

9.1 File List

Here is a list of all documented files with brief descriptions:

CAINode.h	Header containing all the nodes used by the AI	199
Pathfinding.cpp	All the necessary functions to help any AI to traverse any level	200
Pathfinding.h	Class that handles all the necessary functions and variables for the AI to navigate through any level	200
CComponent.h	Fundamental component class of the engine	201
CEntity.h	Fundamental class of the engine with a world transform and ability to have components	203
CAnimationSpriteComponent.h	Extends CSpriteComponent to automatically animate sprite sheets	204
CAudioEmitterComponent.h	Allows a entity to emit audio	205
CCameraComponent.h	Used to attach a camera to a entity	206
CParticleEmitter.h	Allows a entity to emit particles	208
CRigidBodyComponent.cpp	Adds basic rigid body physics to a entity	209
CRigidBodyComponent.h	210
CSpriteComponent.h	A component for loading and displaying a 2D texture in world space as part of CEntity	210
CTextRenderComponent.h	A component for rendering text to the screen from a sprite-sheet	211
Engine.h	213
CParticle.cpp	A helper class for the ParticleEmitter, encapsulates a singlar particle that is emitted	214
CParticle.h	214
CGridCursor.h	215
CTile.h	215
CWorld.h	216
CWorld_Edit.h	218
IInputable.h	221

CCamera.h	Class for storing all camera information needed for rendering	221
CMaterial.h	Holds the directx stuff for uploading sprite specific data to the shader	222
CMesh.h	Holds all information about a mesh for use by CSpriteComponent	223
CTexture.h	Holds all information about a texture for use by CSpriteComponent	224
structures.h	224
CWidget.cpp	Base class for all UI widgets	225
CWidget.h	225
CWidget_Button.cpp	Button Widget class, provides all functionality for buttons and allows to functions to be bound to button events	225
CWidget_Button.h	226
CWidget_Canvas.h	Main container for all widget classes	227
CWidget_Image.h	Image widget class	228
CWidget_Text.h	229
AssetManager.h	A asset manager that holds assets to be retrieved	229
AudioController.h	Internal Audio Controller for the engine	230
CAudio.h	Helper class that encapsulates audio parameters for the audio system	232
CEmitter.h	A helper class to help encapsulate emitters that can be used by the audio system	232
CameraManager.h	Manages the cameras in the engine	233
CollisionComponent.h	234
CTransform.h	A transform class that contains getters and setters	235
CUIManager.h	236
CWorldManager.h	236
Debug.h	Allows for debug logging to a in-game console using ImGui	237
DebugOutput.h	240
EntityManager.h	Static class for tracking entities and components while accommodating translucency	241
EventSystem.h	A generic event system to allow for code to execute across the engine without direct references	242
InputManager.cpp	All the functions needed for the Input Manager	243
InputManager.h	Header containing all the functions and variables needed for the Input Manager	244
IO.h	A Utility class to make IO easier to use	246
Math.h	Utility Math Class	247
Vector3.h	248
Cerberus/Resource.h	252
Necrodoggiecon/Resource.h	252
CT_EditorEntity.h	253
CT_EditorGrid.h	255
CT_EditorMain.h	256
CT_EditorWindows.h	256

WorldConstants.h	257
CerberusTools/CursorEntity.h	258
Necrodoggiecon/Game/CursorEntity.h	258
CWorld_Game.h	259
CWorld_Menu.h	259
AlarmEnemy.cpp	
File containing all the functions needed for the alarm enemy	259
AlarmEnemy.h	
Header file for the alarm enemy	260
CAIController.cpp	
All the functions needed to control the AI	260
CAIController.h	
Header file containing all the functions and variables needed to control the AI	261
DogEnemy.cpp	
File containing all the functions needed for the dog enemy	263
DogEnemy.h	
Header for the dog enemy type	264
GruntEnemy.cpp	
All the functions needed to control the Melee Enemies	265
GruntEnemy.h	
Header file containing all the inherited functions from CAIController and variables needed to control the Melee Enemies	265
State.cpp	
Functions for all the functions for the states	266
State.h	
Header files containing the base state class and any inheritted states for the FSM of the AI	266
AudioEmitterEntity.cpp	
An entity that contains an audio emitter	268
AudioEmitterEntity.h	269
CCharacter.cpp	
Base class for Characters	269
CCharacter.h	269
CInteractable.h	
Entity that can be interacted with	270
CPlayer.h	271
CPlayerController.cpp	
Base class for PlayerControllers, handles functionality for possessing and unpossessing characters	271
CPlayerController.h	272
Dialogue.h	272
DialogueHandler.cpp	
Static class used to control dialogue, including the loading of dialogue from a json	273
DialogueHandler.h	273
DialogueUI.cpp	
Class that stores the UI data for dialogue as well as this, it displays it correctly	273
DialogueUI.h	274
IUsePickup.h	274
LevelTransporter.h	275
NecrodoggieconPage.h	275
PlayerCharacter.h	275
PlayerController.h	276
SoundManager.cpp	
Static class used to handle the playing of audio within the game	276
SoundManager.h	277
TestUI.h	277
WeaponInterface.h	
Interface class to implement the Weapons system using a Strategy Design Strategy	278

WeaponPickup.h	
A class that inherits from CInteractable which allows for weapons to be spawned within the world and picked up by the player	279
weapons.h	
Base Weapon class for the weapons in the game, this will be inherited by the custom classes of the weapons	281
HomingProjectile.cpp	
All the functions needed for Homing Projectile	283
HomingProjectile.h	
Header containing all the functions and variables needed for Homing Projectile	283
LevelCompleteMenu.cpp	
C++ for setting up the level complete screen	284
LevelCompleteMenu.h	
Header for the level complete screen	285
LevelSelectMenu.cpp	
The c++ for the level select menu	285
LevelSelectMenu.h	
Header for the level select menu	286
MainMenu.cpp	
The c++ for the main menu	287
MainMenu.h	
Header for the main menu	288
PauseMenu.cpp	
The c++ for the pause menu	289
PauseMenu.h	
Header for the pause menu	289
Projectile.cpp	
All the functions needed for the Projectile	290
Projectile.h	
Header containing all the functions and variables needed for the Projectile	290
SettingsMenu.cpp	
The c++ for the settings menu	292
SettingsMenu.h	
Header for the settings menu	292
Dagger.h	
Sub-Class for the Dagger weapon	293
Longsword.h	
Sub-Class for the Longsword weapon	294
Rapier.h	
Sub-Class for the Rapier weapon	295
MeleeWeapon.cpp	
Base Melee Weapon class that all Sub-Classes of melee weapons inherit from	296
MeleeWeapon.h	
.	296
Pickup.cpp	
Class to handle scroll pickups	296
Pickup.h	
.	297
InvisibilityScroll.h	
.	297
ShieldScroll.h	
.	297
Crossbow.cpp	
All the functions needed for Crossbow	297
Crossbow.h	
Header containing all the functions and variables needed for Crossbow	298
Fireball.cpp	
All the functions needed for fireball	299
Fireball.h	
Header containing all the functions and variables needed for FireBall	299
MagicMissile.cpp	
All the functions needed for Magic Missile	300

MagicMissile.h	
Header containing all the functions and variables needed for the Magic Missile	300
RangeWeapon.h	301
weaponUI.cpp	
This is the CPP for the weapon UI and the timer	301
weaponUI.h	
Header file for the weapon UI	302

Chapter 10

Class Documentation

10.1 `_Material` Struct Reference

Public Attributes

- int **UseTexture**
- float **padding1** [3]
- XMUINT2 **textureSize**
- XMUINT2 **textureRect**
- XMFLOAT2 **textureOffset**
- int **translucent**
- float **padding2**
- XMFLOAT4 **tint**

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

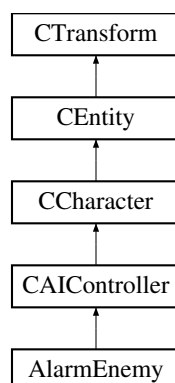
- [CMaterial.h](#)

10.2 `AlarmEnemy` Class Reference

Class for the alarm enemy.

```
#include <AlarmEnemy.h>
```

Inheritance diagram for `AlarmEnemy`:



Public Member Functions

- virtual void [Update](#) (float deltaTime) override
- virtual void [ChasePlayer](#) ([CCharacter](#) *player) override
If not on cooldown then play the bell sound.

Protected Member Functions

- virtual void [OnDeath](#) () override
- virtual void [OnHit](#) (const std::string &hitSound) override

Additional Inherited Members

10.2.1 Detailed Description

Class for the alarm enemy.

It will ring a bell once it sees the player.

10.2.2 Member Function Documentation

10.2.2.1 ChasePlayer()

```
void AlarmEnemy::ChasePlayer (
    CCharacter * player ) [override], [virtual]
```

If not on cooldown then play the bell sound.

Parameters

<i>player</i>	Player that it can see.
---------------	-------------------------

Reimplemented from [CAIController](#).

10.2.2.2 OnDeath()

```
void AlarmEnemy::OnDeath ( ) [override], [protected], [virtual]
```

Reimplemented from [CAIController](#).

10.2.2.3 OnHit()

```
void AlarmEnemy::OnHit (
    const std::string & hitSound ) [override], [protected], [virtual]
```

Reimplemented from [CAIController](#).

10.2.2.4 Update()

```
void AlarmEnemy::Update (
    float deltaTime ) [override], [virtual]
```

Parameters

<i>deltaTime</i>	
------------------	--

Reimplemented from [CAIController](#).

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

- [AlarmEnemy.h](#)
- [AlarmEnemy.cpp](#)

10.3 AssetManager Class Reference

Static Public Member Functions

- static [CMesh](#) * [AddMesh](#) (std::string meshID, [CMesh](#) *mesh)
Adds a [CMesh](#) to the asset manager.
- static [CMesh](#) * [GetMesh](#) (std::string meshID)
Returns the mesh in the asset manager if it exists.
- static [CMesh](#) * [GetDefaultMesh](#) ()
Returns the default mesh held within the asset manager.
- static [CTexture](#) * [GetTexture](#) (std::string texturePath)
Returns a texture at a specified texture path.
- static [CTexture](#) * [GetTextureWIC](#) (std::string texturePath)
Returns a texture at a specified texture path.
- static [CAudio](#) * [AddAudio](#) (std::string audioPath, [CAudio](#) *audio)
Adds a audio clip to the asset manager.
- static [CAudio](#) * [GetAudio](#) (std::string audioPath)
Returns a stored audio at a path.
- static void [RemoveAudio](#) (std::string audioPath)
Removes a audio from the asset manager.
- static void [Destroy](#) ()
Destroys the asset manager.

10.3.1 Member Function Documentation

10.3.1.1 AddAudio()

```
CAudio * AssetManager::AddAudio (
    std::string audioPath,
    CAudio * audio ) [static]
```

Adds a audio clip to the asset manager.

Parameters

<i>audioPath</i>	the audio path you wish to add
<i>audio</i>	a pointer to the audio that you wish to store.

Returns

returns a pointer to the stored audio.

10.3.1.2 AddMesh()

```
CMesh * AssetManager::AddMesh (
    std::string meshID,
    CMesh * mesh ) [static]
```

Adds a CMesh to the asset manager.

Parameters

<i>meshID</i>	the meshID that is used to retrieve the mesh later.
<i>mesh</i>	the mesh that you wish to store.

Returns

CMesh pointer to the stored mesh.

10.3.1.3 GetAudio()

```
CAudio * AssetManager::GetAudio (
    std::string audioPath ) [static]
```

Returns a stored audio at a path.

Parameters

<i>audioPath</i>	the path of the audio you wish to retrieve.
------------------	---

Returns

a pointer to the retrieved audio.

10.3.1.4 GetDefaultMesh()

```
CMesh * AssetManager::GetDefaultMesh ( ) [static]
```

Returns the default mesh held within the asset manager.

Returns

the default mesh held within the manager

10.3.1.5 GetMesh()

```
CMesh * AssetManager::GetMesh (
    std::string meshID ) [static]
```

Returns the mesh in the asset manager if it exists.

Parameters

<i>meshID</i>	the meshID of the mesh you wish to retrieve.
---------------	--

Returns

a pointer to the mesh that was retrieved.

10.3.1.6 GetTexture()

```
CTexture * AssetManager::GetTexture (
    std::string texturePath ) [static]
```

Returns a texture at a specified texture path.

Parameters

<i>texturePath</i>	the texture path that you wish to retrieve.
--------------------	---

Returns

a pointer to the retrieved texture.

10.3.1.7 GetTextureWIC()

```
CTexture * AssetManager::GetTextureWIC (
    std::string texturePath ) [static]
```

Returns a texture at a specified texture path.

Parameters

<i>texturePath</i>	the texture path that you wish to retrieve.
--------------------	---

Returns

a pointer to the retrieved texture.

10.3.1.8 RemoveAudio()

```
void AssetManager::RemoveAudio (
    std::string audioPath ) [static]
```

Removes a audio from the asset manager.

Parameters

<i>audioPath</i>	the audio path that you wish to remove.
------------------	---

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

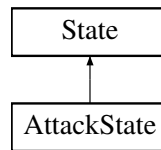
- [AssetManager.h](#)
- [AssetManager.cpp](#)

10.4 AttackState Class Reference

[State](#) for when the AI is attacking the player.

```
#include <State.h>
```

Inheritance diagram for AttackState:



Public Member Functions

- void [Enter](#) ([CAIController](#) *controller) override
- void [Update](#) ([CAIController](#) *controller, float deltaTime) override
- void [Exit](#) ([CAIController](#) *controller) override

Static Public Member Functions

- static [State](#) & [getInstance](#) ()

10.4.1 Detailed Description

[State](#) for when the AI is attacking the player.

10.4.2 Member Function Documentation

10.4.2.1 Enter()

```
void AttackState::Enter (  
    CAIController * controller ) [override], [virtual]
```

Reimplemented from [State](#).

10.4.2.2 Exit()

```
void AttackState::Exit (  
    CAIController * controller ) [override], [virtual]
```

Reimplemented from [State](#).

10.4.2.3 Update()

```
void AttackState::Update (
    CAIController * controller,
    float deltaTime ) [override], [virtual]
```

Reimplemented from [State](#).

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

- [State.h](#)
- [State.cpp](#)

10.5 AudioController Class Reference

Static Public Member Functions

- static void **Initialize** ()
Initializes the audio system and FMOD.
- static void **Shutdown** ()
Shutsdown the audio system and FMOD.
- static [CAudio](#) * **LoadAudio** (const std::string &path)
Loads a audio into FMOD and the audio system.
- static bool **PlayAudio** (const std::string &path)
Plays a audio using FMOD.
- static bool **PlayAudio** (const std::string &path, bool loop)
Plays a audio using FMOD with the ability to loop.
- static bool **StopAudio** (const std::string &path)
Stops a audio from playing.
- static bool **DestroyAudio** (const std::string &path)
Deletes a audio from FMOD and the audio system.
- static void **Update** (float deltaTime)
Updates the overall audio volume to simulate 3D audio.
- static std::vector< [CEmitter](#) * > **GetAllEmittersWithinRange** ([Vector3](#) position, bool checkIfPlaying)
Returns all emitters within range of a position.
- static bool **AddEmitter** ([CEmitter](#) *emitter)
Adds a emitter to the audio system.
- static bool **RemoveEmitter** ([CEmitter](#) *emitter)
Removes a emitter from the audio system.
- static void **SetMaxVolumeForEmitterType** (const float volume, EMITTERTYPE type)
- static bool **AddListener** ([CTransform](#) *listenerPos)
Adds a listener to the audio controller, used for attenuation.

10.5.1 Member Function Documentation

10.5.1.1 AddEmitter()

```
bool AudioController::AddEmitter (
    CEmitter * emitter ) [static]
```

Adds a emitter to the audio system.

Parameters

<i>emitter</i>	emitter you wish to add to the audio system.
----------------	--

Returns

bool on success or failure

10.5.1.2 AddListener()

```
bool AudioController::AddListener (
    CTransform * listenerPos ) [static]
```

Adds a listener to the audio controller, used for attenuation.

Parameters

<i>listenerPositon</i>	the position of the listener that controls the attenuation of the audio system.
------------------------	---

Returns

bool on success or failure

10.5.1.3 DestroyAudio()

```
bool AudioController::DestroyAudio (
    const std::string & path ) [static]
```

Deletes a audio from FMOD and the audio system.

Parameters

<i>path</i>	to audio that you wish to destroy
-------------	-----------------------------------

Returns

bool on success or failure

10.5.1.4 GetAllEmittersWithinRange()

```
std::vector< CEmitter * > AudioController::GetAllEmittersWithinRange (
    Vector3 position,
    bool checkIfPlaying ) [static]
```

Returns all emitters within range of a position.

Parameters

<i>position</i>	sampling position, should be at the center of the search area.
-----------------	--

Returns

a vector of emitters that where in range and satisfied the argument conditions.

10.5.1.5 LoadAudio()

```
CAudio * AudioController::LoadAudio (
    const std::string & path ) [static]
```

Loads a audio into FMOD and the audio system.

Parameters

<i>path</i>	to audio you wish to load.
-------------	----------------------------

Returns

CAudio pointer to the created audio.

10.5.1.6 PlayAudio() [1/2]

```
bool AudioController::PlayAudio (
    const std::string & path ) [static]
```

Plays a audio using FMOD.

Parameters

<i>path</i>	to audio you wish to play.
-------------	----------------------------

Returns

bool on success or failure.

10.5.1.7 PlayAudio() [2/2]

```
bool AudioController::PlayAudio (
    const std::string & path,
    bool loop ) [static]
```

Plays a audio using FMOD with the ability to loop.

Parameters

<i>path</i>	to audio you wish to play
<i>loop</i>	whether you would like the audio to loop.

Returns

bool on success or failure.

10.5.1.8 RemoveEmitter()

```
bool AudioController::RemoveEmitter (
    CEmitter * emitter ) [static]
```

Removes a emitter from the audio system.

Parameters

<i>emitter</i>	emitter you wish to add to the audio system.
----------------	--

Returns

bool on success or failure

10.5.1.9 StopAudio()

```
bool AudioController::StopAudio (
    const std::string & path ) [static]
```

Stops a audio from playing.

Parameters

<i>path</i>	to audio you wish to stop playing.
-------------	------------------------------------

Returns

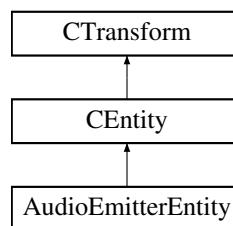
bool on success or failure

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

- [AudioController.h](#)
- [AudioController.cpp](#)

10.6 AudioEmitterEntity Class Reference

Inheritance diagram for AudioEmitterEntity:



Public Member Functions

- void [SetAudio](#) (const std::string &audioPath, float range)
Function to set the audio that the emitter should store.
- void [SetAudio](#) (const std::string &audioPath, float range, bool ambient)
Function to set the audio that the emitter should store.
- void [PlayAudio](#) ([Vector3](#) position)
Function to play the stored audio at the appropriate position.
- void **Stop** ()
Function to stop the audio emitter from playing.
- void [PlayAudio](#) (const std::string &audioPath)
Function to load and play audio from a file path.
- void [PlayAudio](#) (bool shouldLoop)
Function to play the stored audio.
- void [Load](#) (const std::string &audioPath, bool ambient)
Function to load audio from a file.
- void [SetRange](#) (float range)
Function to set the range of the audio.
- void **SetAttachedEntity** ([CEntity](#) *entity)

Protected Member Functions

- virtual void [Update](#) (float deltaTime) override
Function inherited from [CEntity](#).

Protected Attributes

- [CAudioEmitterComponent](#) * **audioEmitter**
- [CEntity](#) * **attachedEntity**
- bool **isAttached**

Additional Inherited Members

10.6.1 Member Function Documentation

10.6.1.1 Load()

```
void AudioEmitterEntity::Load (
    const std::string & audioPath,
    bool ambient )
```

Function to load audio from a file.

Parameters

<i>audioPath</i>	- Path to the audio file
<i>ambient</i>	- Whether or not the audio is ambient

10.6.1.2 PlayAudio() [1/3]

```
void AudioEmitterEntity::PlayAudio (
    bool shouldLoop )
```

Function to play the stored audio.

Parameters

<i>shouldLoop</i>	- Whether or not the audio should loop
-------------------	--

10.6.1.3 PlayAudio() [2/3]

```
void AudioEmitterEntity::PlayAudio (
    const std::string & audioPath )
```

Function to load and play audio from a file path.

Parameters

<i>audioPath</i>	- Path to the audio file
------------------	--------------------------

10.6.1.4 PlayAudio() [3/3]

```
void AudioEmitterEntity::PlayAudio (
    Vector3 position )
```

Function to play the stored audio at the appropriate position.

Parameters

<i>position</i>	- The position to play the audio at
-----------------	-------------------------------------

10.6.1.5 SetAudio() [1/2]

```
void AudioEmitterEntity::SetAudio (
    const std::string & audioPath,
    float range )
```

Function to set the audio that the emitter should store.

Parameters

<i>audioPath</i>	- Path to the audio file
<i>range</i>	- The range of the audio

10.6.1.6 SetAudio() [2/2]

```
void AudioEmitterEntity::SetAudio (
    const std::string & audioPath,
    float range,
    bool ambient )
```

Function to set the audio that the emitter should store.

Parameters

<i>audioPath</i>	- Path to the audio file
<i>range</i>	- The range of the audio
<i>ambient</i>	- Whether the audio is ambient or not

10.6.1.7 SetRange()

```
void AudioEmitterEntity::SetRange (
    float range )
```

Function to set the range of the audio.

Parameters

<i>range</i>	- The new range for the audio emitter
--------------	---------------------------------------

10.6.1.8 Update()

```
void AudioEmitterEntity::Update (
    float deltaTime ) [override], [protected], [virtual]
```

Function inherited from [CEntity](#).

Used to ensure the Entity follows the attached entities position

Parameters

<i>deltaTime</i>	- Time since the last frame
------------------	-----------------------------

Implements [CEntity](#).

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

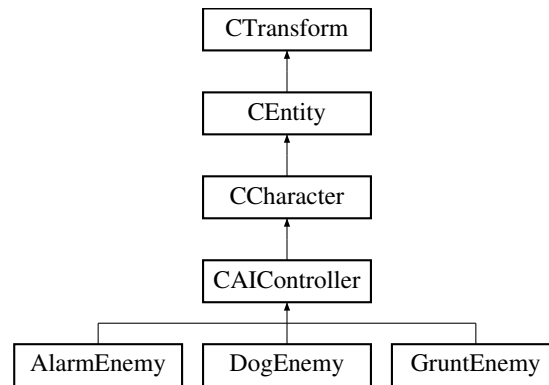
- [AudioEmitterEntity.h](#)
- [AudioEmitterEntity.cpp](#)

10.7 CAIController Class Reference

Controller class for the AI.

```
#include <CAIController.h>
```

Inheritance diagram for CAIController:



Public Member Functions

- void **SetRotationSpeed** (float speed)
- float **GetRotationSpeed** ()
- void **SetSearchTime** (float time)
- float **GetSearchTime** ()
- void **SetInitialSpeed** (float speed)
- float **GetInitialSpeed** ()
- void **SetSpeed** (float speed)
- float **GetSpeed** ()
- void **SetMass** (float mass)
- float **GetMass** ()
- void **SetRange** (float range)
- float **GetRange** ()
- void **SetViewAngle** (float angle)
- float **GetViewAngle** ()
- void **SetWidth** (float wide)
- float **GetWidth** ()
- void **SetHeight** (float high)
- float **GetHeight** ()
- void **SetPositionToInvestigate** ([Vector3](#) pos)
- [Vector3](#) **GetPositionToInvestigate** ()
- void **SetIsAttacking** (bool isAttack)
- bool **GetIsAttacking** ()
- void **SetSpriteSize** (float size)
- float **GetSpriteSize** ()
- void **SetIsBoss** (bool boss)
- bool **GetIsBoss** ()
- virtual void [Update](#) (float deltaTime) override
- void **Patrolling** ()
 - Moves the direction of the character towards the next point in the path.*
- void **SearchForPlayer** ()
 - Spin on the spot trying to find the player.*
- void **Investigating** ([Vector3](#) positionOfInterest)
 - Moves the AI along the path to the position of interest.*
- virtual void [AttackEnter](#) ([CCharacter](#) *player)
- virtual void [ChaseEnter](#) ()
 - Enter function for the chase state.*
- virtual void [ChasePlayer](#) ([CCharacter](#) *player)
 - Seek towards the player and if it gets close then switch to the attacking state.*

- virtual void **AttackPlayer** (CCharacter *player, float deltaTime)
Attack the player using the weapon attached.
- void **SetCurrentState** (State &state)
Exits one state and enters the state passed in.
- bool **CanSee** (CCharacter *player)
Maths magic that determines whether the player is in view.
- void **SetPathNodes** (std::vector< WaypointNode * > nodes)
Sets the path nodes for the AI.
- void **SetPath** ()
Sets the path between the closest waypoint to the character and the closest waypoint to the target patrol node.
- void **SetPath** (Vector3 endPosition)
Sets the path between the closest waypoint to the AI and the closest waypoint to the end position.
- void **ApplyDamage** (float damageAmount)
Apply damage to the enemy.
- void **ApplyDamage** (float damageAmount, const std::string &hitAudioPath)

Public Attributes

- Pathfinding * **pathing**
- class CAnimationSpriteComponent * **sprite** = nullptr

Protected Member Functions

- virtual void **OnHit** (const std::string &hitSound)
- virtual void **OnDeath** ()
- void **Movement** (float deltaTime)
Moves the character position using acceleration, force, mass and velocity.
- **Vector3 CollisionAvoidance** ()
Finds the closest obstacle and calculates the vector to avoid it.
- **Vector3 Seek** (Vector3 TargetPos)
Returns the velocity change needed to reach the target position.
- void **CheckForPlayer** ()
Checks if the player is in view.
- void **MoveViewFrustrum** ()
Moves the view frustrum attached to the AI.
- virtual void **HasCollided** (CollisionComponent *collidedObject)

Protected Attributes

- class CSpriteComponent * **viewFrustrum** = nullptr
- **Vector3 positionToInvestigate**
- **Vector3 velocity**
- **Vector3 acceleration**
- **Vector3 heading**
- **Vector3 aiPosition**
- std::vector< CTile * > **tiles**
- std::vector< CTile * > **obstacles**
- **PatrolNode** * **currentPatrolNode**
- std::vector< WaypointNode * > **pathNodes**
- int **currentCount**

- bool **isAttacking** = false
- bool **isBoss** = false
- [CCharacter](#) * **playerToKill** = nullptr
- [CCharacter](#) * **playerToChase** = nullptr
- [Vector3](#) **originalViewFrustrumPosition**
- std::vector< [CCharacter](#) * > **characters** = Engine::GetEntityOfType<[CCharacter](#)>()
- std::vector< [CCharacter](#) * > **players**
- float **aiSpeed** = 100.0f
- float **initialSpeed** = aiSpeed
- float **aiMass** = 10.0f
- float **aiRange** = 400.0f
- float **aiViewAngle** = 90.0f
- float **width** = 64.0f
- float **height** = 64.0f
- float **rotationSpeed** = 0.01f
- float **maxSearchTime** = 5.0f
- float **searchTimer** = 0.0f
- float **sizeOfTiles** = 0.0f
- float **spriteSize** = 64.0f
- [State](#) * **currentState**

10.7.1 Detailed Description

Controller class for the AI.

10.7.2 Member Function Documentation

10.7.2.1 ApplyDamage() [1/2]

```
void CAIController::ApplyDamage (
    float damageAmount ) [virtual]
```

Apply damage to the enemy.

Parameters

<i>damageAmount</i>	Amount to damage the enemy.
<i>damageCauser</i>	Root of the damage.

Reimplemented from [CCharacter](#).

10.7.2.2 ApplyDamage() [2/2]

```
void CAIController::ApplyDamage (
    float damageAmount,
    const std::string & hitAudioPath ) [virtual]
```

Reimplemented from [CCharacter](#).

10.7.2.3 AttackEnter()

```
void CAIController::AttackEnter (
    CCharacter * player ) [virtual]
```

Reimplemented in [DogEnemy](#).

10.7.2.4 AttackPlayer()

```
void CAIController::AttackPlayer (
    CCharacter * player,
    float deltaTime ) [virtual]
```

Attack the player using the weapon attached.

Parameters

<i>player</i>	Player to attack.
---------------	-------------------

Reimplemented in [DogEnemy](#), and [GruntEnemy](#).

10.7.2.5 CanSee()

```
bool CAIController::CanSee (
    CCharacter * player )
```

Maths magic that determines whether the player is in view.

Parameters

<i>posOfObject</i>	Vector3 representing the position of the object to see.
--------------------	---

Returns

Returns a boolean determining whether the object is in view.

10.7.2.6 ChaseEnter()

```
void CAIController::ChaseEnter ( ) [virtual]
```

Enter function for the chase state.

Called once when first switching to this state.

10.7.2.7 ChasePlayer()

```
void CAIController::ChasePlayer (
    CCharacter * player ) [virtual]
```

Seek towards the player and if it gets close then switch to the attacking state.

Reimplemented in [AlarmEnemy](#), [DogEnemy](#), and [GruntEnemy](#).

10.7.2.8 CollisionAvoidance()

```
Vector3 CAIController::CollisionAvoidance ( ) [protected]
```

Finds the closest obstacle and calculates the vector to avoid it.

Returns

Returns a Vector3 that is the direction to avoid the obstacle.

10.7.2.9 HasCollided()

```
virtual void CAIController::HasCollided (
    CollisionComponent * collidedObject ) [inline], [protected], [virtual]
```

Reimplemented from [CEntity](#).

10.7.2.10 Investigating()

```
void CAIController::Investigating (
    Vector3 positionOfInterest )
```

Moves the AI along the path to the position of interest.

Parameters

<i>positionOfInterest</i>	Position for the AI to investigate.
---------------------------	-------------------------------------

10.7.2.11 Movement()

```
void CAIController::Movement (
    float deltaTime ) [protected]
```

Moves the character position using acceleration, force, mass and velocity.

Parameters

<i>deltaTime</i>	Time between frames.
<i>deltaTime</i>	

10.7.2.12 Seek()

```
Vector3 CAIController::Seek (
    Vector3 TargetPos ) [protected]
```

Returns the velocity change needed to reach the target position.

Parameters

<i>TargetPos</i>	Vector3 representing the position for the AI to go.
------------------	---

Returns

Returns the direction to the target position.

10.7.2.13 SetCurrentState()

```
void CAIController::SetCurrentState (
    State & state )
```

Exits one state and enters the state passed in.

Parameters

<i>state</i>	State to switch to.
--------------	---------------------

10.7.2.14 SetPath()

```
void CAIController::SetPath (
    Vector3 endPosition )
```

Sets the path between the closest waypoint to the AI and the closest waypoint to the end position.

Parameters

<i>endPosition</i>	Target position for the end of the path.
--------------------	--

10.7.2.15 SetPathNodes()

```
void CAIController::SetPathNodes (
    std::vector< WaypointNode * > nodes )
```

Sets the path nodes for the AI.

Parameters

<i>nodes</i>	Vector array of waypoint nodes to set.
--------------	--

10.7.2.16 Update()

```
void CAIController::Update (
    float deltaTime ) [override], [virtual]
```

Parameters

<i>deltaTime</i>	
------------------	--

Reimplemented from [CCharacter](#).

Reimplemented in [AlarmEnemy](#), [DogEnemy](#), and [GruntEnemy](#).

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

- [CAIController.h](#)
- [CAIController.cpp](#)

10.8 CameraManager Class Reference

Static Public Member Functions

- static void [AddCamera](#) ([CCameraComponent](#) *camera)
Adds a camera to the manager.
- static void [RemoveCamera](#) ([CCameraComponent](#) *camera)

- Removes a camera from the manager.*
- static `CCameraComponent * GetRenderingCamera ()`
Returns the rendering camera.
- static void `SetRenderingCamera (CCameraComponent *camera)`
Sets the rendering camera.
- static `std::vector< CCameraComponent * > GetAllCameras ()`
Returns a vector of all cameras inside the manager.

10.8.1 Member Function Documentation

10.8.1.1 AddCamera()

```
void CameraManager::AddCamera (
    CCameraComponent * camera ) [static]
```

Adds a camera to the manager.

Parameters

<i>camera</i>	camera you wish to add.
---------------	-------------------------

10.8.1.2 GetAllCameras()

```
std::vector< CCameraComponent * > CameraManager::GetAllCameras ( ) [static]
```

Returns a vector of all cameras inside the manager.

Returns

a vector of all cameras stored within the camera manager.

10.8.1.3 GetRenderingCamera()

```
CCameraComponent * CameraManager::GetRenderingCamera ( ) [static]
```

Returns the rendering camera.

Returns

the current rendering camera.

10.8.1.4 RemoveCamera()

```
void CameraManager::RemoveCamera (
    CCameraComponent * camera ) [static]
```

Removes a camera from the manager.

Further, if a rendering camera is delete it will move the rendering camera to the next camera in the manager.

Parameters

<i>camera</i>	camera you wish to remove.
---------------	----------------------------

10.8.1.5 SetRenderingCamera()

```
void CameraManager::SetRenderingCamera (
    CCameraComponent * camera ) [static]
```

Sets the rendering camera.

Parameters

<i>camera</i>	the camera you wish to set as the rendering camera.
---------------	---

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

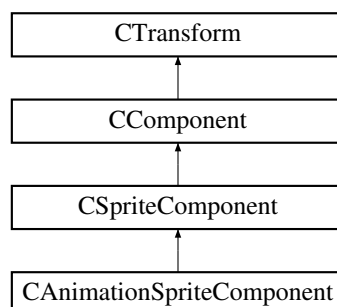
- [CameraManager.h](#)
- CameraManager.cpp

10.9 CAnimationSpriteComponent Class Reference

Extends [CSpriteComponent](#) to automatically animate sprite-sheets.

```
#include <CAnimationSpriteComponent.h>
```

Inheritance diagram for CAnimationSpriteComponent:



Public Member Functions

- void **ResetAnimation** ()
- void **SetAnimationRectSize** (const XMUINT2 &newSize, const bool &resetAnimation=false)
Sets the size of the rectangle in sprites to which the animation is played within.
- const XMUINT2 & **GetAnimationRectSize** ()
- void **SetAnimationRectPosition** (const XMUINT2 &newPosition, const bool &resetAnimation=false)
Sets the position of the rectangle in sprites to which the animation is played within.
- const XMUINT2 & **GetAnimationRectPosition** ()
- const XMUINT2 & **GetCurrentFrame** ()
- void **SetPlaying** (const bool &newState, const bool &resetAnimation=false)
Set if the animation should be playing.
- const bool & **GetPlaying** ()
- void **SetElapsedTime** (const float &newTime)
Set the current animation time in the form of elapsed time.
- const float & **GetElapsedTime** ()
- void **SetAnimationSpeed** (const float &newSpeed)
Sets the speed of the animation in frames per second - Default 24.
- const float & **GetAnimationSpeed** ()
- virtual void **Update** (float deltaTime) override
Updated automatically every single frame.

Additional Inherited Members

10.9.1 Detailed Description

Extends [CSpriteComponent](#) to automatically animate sprite-sheets.

10.9.2 Member Function Documentation

10.9.2.1 SetAnimationRectPosition()

```
void CAnimationSpriteComponent::SetAnimationRectPosition (
    const XMUINT2 & newPosition,
    const bool & resetAnimation = false ) [inline]
```

Sets the position of the rectangle in sprites to which the animation is played within.

This is the point of the top left of the animation rect. Use this to select the portion of the sprite to animate.

10.9.2.2 SetAnimationRectSize()

```
void CAnimationSpriteComponent::SetAnimationRectSize (
    const XMUINT2 & newSize,
    const bool & resetAnimation = false ) [inline]
```

Sets the size of the rectangle in sprites to which the animation is played within.

Like narrowing down the sprite to just the animation you want.

10.9.2.3 Update()

```
void CAnimationSpriteComponent::Update (
    float deltaTime ) [override], [virtual]
```

Updated automatically every single frame.

Reimplemented from [CSpriteComponent](#).

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

- [CAnimationSpriteComponent.h](#)
- [CAnimationSpriteComponent.cpp](#)

10.10 CAudio Class Reference

Public Member Functions

- **CAudio** (std::string path, FMOD::Sound *sound, FMOD::ChannelGroup *group)
- **CAudio** (std::string path, FMOD::Sound *sound, FMOD::ChannelGroup *group, FMOD::Channel *channel)

Public Attributes

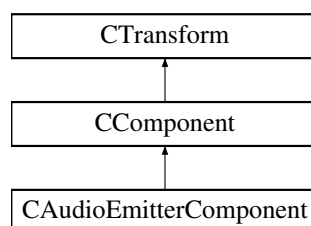
- std::string **path**
- FMOD::Sound * **sound**
- FMOD::ChannelGroup * **group**
- FMOD::Channel * **channel**
- float **maxVolume**

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

- [CAudio.h](#)

10.11 CAudioEmitterComponent Class Reference

Inheritance diagram for CAudioEmitterComponent:



Public Member Functions

- void [Load](#) (const std::string &path)
Loads a audio to be used by the emitter.
- void [Load](#) (const std::string &path, bool ambient)
Loads a audio to be used by the emitter.
- void [Play](#) ()
Plays the audio emitter.
- void [Play](#) (bool loop)
Plays the audio emitter with a option of looping the audio.
- void [Stop](#) ()
Stops the audio emitter.
- void [SetRange](#) (float range)
Sets the range at which the audio can be heard.
- virtual void [Update](#) (float deltaTime)
Updates the audio emitters position.
- virtual void [Draw](#) (struct ID3D11DeviceContext *context, const XMFLOAT4X4 &parentMat, [ConstantBuffer](#) cb, ID3D11Buffer *constantBuffer)
Almost the same as [Update\(\)](#) but to be used for drawing only.

Additional Inherited Members

10.11.1 Member Function Documentation

10.11.1.1 Draw()

```
virtual void CAudioEmitterComponent::Draw (
    struct ID3D11DeviceContext * context,
    const XMFLOAT4X4 & parentMat,
    ConstantBuffer cb,
    ID3D11Buffer * constantBuffer ) [inline], [virtual]
```

Almost the same as [Update\(\)](#) but to be used for drawing only.

Implements [CComponent](#).

10.11.1.2 Load() [1/2]

```
void CAudioEmitterComponent::Load (
    const std::string & path )
```

Loads a audio to be used by the emitter.

Parameters

<i>path</i>	path to audio
-------------	---------------

10.11.1.3 Load() [2/2]

```
void CAudioEmitterComponent::Load (  
    const std::string & path,  
    bool ambient )
```

Loads a audio to be used by the emitter.

Parameters

<i>path</i>	path to audio
-------------	---------------

10.11.1.4 Play()

```
void CAudioEmitterComponent::Play (  
    bool loop )
```

Plays the audio emitter with a option of looping the audio.

Parameters

<i>loop</i>	
-------------	--

10.11.1.5 SetRange()

```
void CAudioEmitterComponent::SetRange (  
    float range )
```

Sets the range at which the audio can be heard.

Parameters

<i>range</i>	hearing distance of audio.
--------------	----------------------------

10.11.1.6 Update()

```
void CAudioEmitterComponent::Update (
    float deltaTime ) [virtual]
```

Updates the audio emitters position.

Parameters

<i>deltaTime</i>	
------------------	--

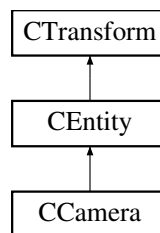
Implements [CComponent](#).

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

- [CAudioEmitterComponent.h](#)
- [CAudioEmitterComponent.cpp](#)

10.12 CCamera Class Reference

Inheritance diagram for CCamera:



Public Member Functions

- virtual void [Update](#) (float *deltaTime*)
Updated automatically every single frame.

Additional Inherited Members

10.12.1 Member Function Documentation

10.12.1.1 Update()

```
virtual void CCamera::Update (
    float deltaTime ) [inline], [virtual]
```

Updated automatically every single frame.

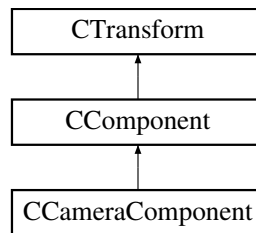
Implements [CEntity](#).

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

- [CCamera.h](#)

10.13 CCameraComponent Class Reference

Inheritance diagram for CCameraComponent:



Public Member Functions

- virtual void [Update](#) (float deltaTime) override
Updates the camera's view matrix if the position has changed.
- virtual void [Draw](#) (struct ID3D11DeviceContext *context, const XMFLOAT4X4 &parentMat, [ConstantBuffer](#) cb, ID3D11Buffer *constantBuffer) override
Almost the same as [Update\(\)](#) but to be used for drawing only.
- void [SetZoomLevel](#) (const float level)
Sets the zoom level of the camera (FOV).
- float [GetZoomLevel](#) ()
Returns the zoom level of the camera.
- void [SetAttachedToParent](#) (const bool value)
Sets whether the camera is attached to the parent or if it can move on its own.
- bool [getAttachedToParent](#) ()
Returns whether the camera is attached to the parent or if it can move on its own.
- XMFLOAT4X4 [GetViewMatrix](#) ()
Returns the view matrix of the camera.
- XMFLOAT4X4 [GetProjectionMatrix](#) ()
Returns the projection matrix of the camera.
- [Vector3](#) [GetPosition](#) ()
Returns the position of the camera's parent entity.
- void **UpdateView** ()
Updates the view matrix of the camera.
- void **UpdateProj** ()
Updates the projection matrix of the camera.

Additional Inherited Members

10.13.1 Member Function Documentation

10.13.1.1 Draw()

```
virtual void CCameraComponent::Draw (
    struct ID3D11DeviceContext * context,
    const XMFLOAT4X4 & parentMat,
    ConstantBuffer cb,
    ID3D11Buffer * constantBuffer ) [inline], [override], [virtual]
```

Almost the same as [Update\(\)](#) but to be used for drawing only.

Implements [CComponent](#).

10.13.1.2 getAttachedToParent()

```
bool CCameraComponent::getAttachedToParent ( )
```

Returns whether the camera is attached to the parent or if it can move on its own.

Returns

whether you are attached to your parent or not.

10.13.1.3 GetPosition()

```
Vector3 CCameraComponent::GetPosition ( )
```

Returns the position of the camera's parent entity.

Returns

cameras' parent entity's position.

10.13.1.4 GetProjectionMatrix()

```
XMFLOAT4X4 CCameraComponent::GetProjectionMatrix ( )
```

Returns the projection matrix of the camera.

Returns

projection-matrix of camera.

10.13.1.5 GetViewMatrix()

```
XMFLLOAT4X4 CCameraComponent::GetViewMatrix ( )
```

Returns the view matrix of the camera.

Returns

view-matrix of camera.

10.13.1.6 GetZoomLevel()

```
float CCameraComponent::GetZoomLevel ( )
```

Returns the zoom level of the camera.

Returns

zoom-level of camera.

10.13.1.7 SetAttachedToParent()

```
void CCameraComponent::SetAttachedToParent (
    const bool value )
```

Sets whether the camera is attached to the parent or if it can move on its own.

Parameters

<i>value</i>	whether you would like for the camera to be attached to the parent or not.
--------------	--

10.13.1.8 SetZoomLevel()

```
void CCameraComponent::SetZoomLevel (
    const float level )
```

Sets the zoom level of the camera (FOV).

Parameters

<i>level</i>	the zoom level you wish for the camera to be.
--------------	---

10.13.1.9 Update()

```
void CCameraComponent::Update (
    float deltaTime ) [override], [virtual]
```

Updates the camera's view matrix if the position has changed.

Parameters

<i>deltaTime</i>	
------------------	--

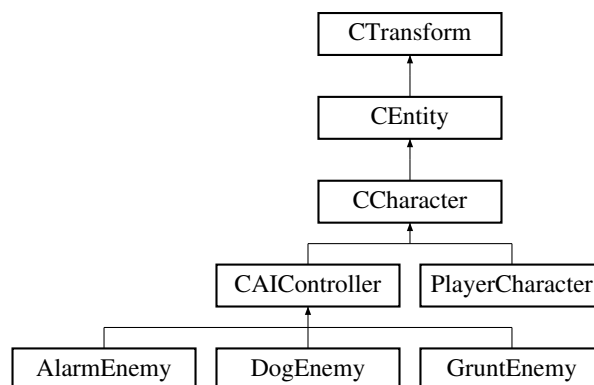
Implements [CComponent](#).

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

- [CCameraComponent.h](#)
- [CCameraComponent.cpp](#)

10.14 CCharacter Class Reference

Inheritance diagram for CCharacter:



Public Member Functions

- virtual void [ApplyDamage](#) (float damageAmount)
Public function used to apply damage to the character.
- virtual void **ApplyDamage** (float damageAmount, const std::string &onHitSound)
- virtual void [Update](#) (float deltaTime)
Updated automatically every single frame.
- void **EquipWeapon** ([Weapon](#) *weapon)
- void **UpdateWeaponSprite** ()
- void **SetHealth** (float heal)
- float **GetHealth** ()
- void **SetIsPlayer** (bool player)
- bool **GetIsPlayer** ()
- bool **GetVisible** ()
- [Weapon](#) * **GetWeapon** ()

Protected Member Functions

- void **UpdateWeaponSpritePosition** ([CSpriteComponent](#) *wSprite)
- void **AddMovement** (XMFLOAT2 vel, float deltaTime)

Protected Attributes

- bool **isPlayer** = false
- bool **visible** = true
- float **health** = 1.0f
- [WeaponInterface](#) * **weaponComponent** = nullptr
- [CSpriteComponent](#) * **weaponSprite** = nullptr

Additional Inherited Members

10.14.1 Member Function Documentation

10.14.1.1 ApplyDamage()

```
virtual void CCharacter::ApplyDamage (  
    float damageAmount ) [inline], [virtual]
```

Public function used to apply damage to the character.

Reimplemented in [PlayerCharacter](#), and [CAIController](#).

10.14.1.2 Update()

```
virtual void CCharacter::Update (  
    float deltaTime ) [inline], [virtual]
```

Updated automatically every single frame.

Implements [CEntity](#).

Reimplemented in [AlarmEnemy](#), [CAIController](#), [DogEnemy](#), [GruntEnemy](#), and [PlayerCharacter](#).

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

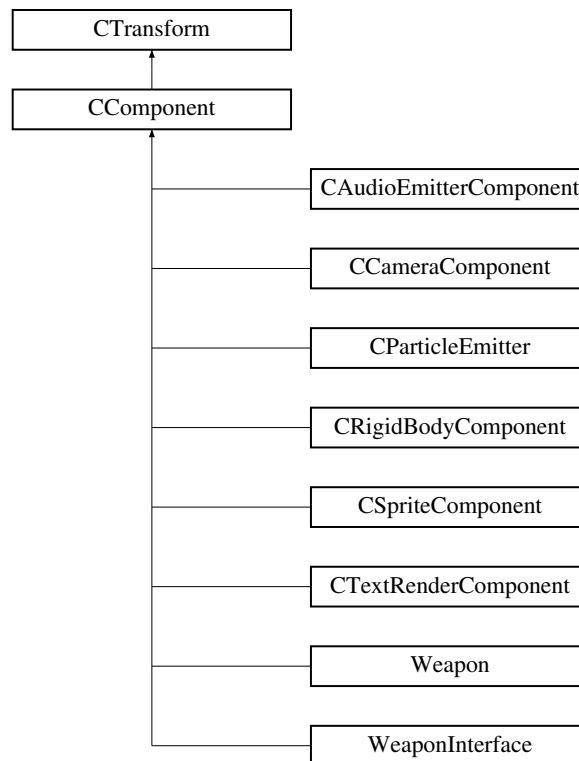
- CCharacter.h
- [CCharacter.cpp](#)

10.15 CComponent Class Reference

Fundamental component class of the engine.

```
#include <CComponent.h>
```

Inheritance diagram for CComponent:



Public Member Functions

- void [SetAnchor](#) (const XMFLOAT2 &newAnchor)
Sets the region of the screen a UI element will be "anchored" to.
- virtual void [SetUseTranslucency](#) (const bool &newTranslucency)
Sets if this component will/can draw translucent pixels.
- void **SetIsUI** (const bool &newIsUI)
Sets if this component will be drawn in world space or screen space.
- void **SetShouldUpdate** (const bool &newShouldUpdate)
Sets if this component will be automatically updated via the [Update\(\)](#).
- void **SetShouldDraw** (const bool &newShouldDraw)
Sets if this component will be automatically drawn via the [Draw\(\)](#).
- void **SetLastResolution** (const XMUINT2 &newLastResolution)
Sets the last resolution variable of the screen for rendering uses.
- void **SetParent** (class [CEntity](#) *newParent)
Set the parent entity of this component, done automatically.
- void **SetName** (const std::string &newName)
Sets the name of the component mostly for debugging purposes.
- const bool & **GetShouldUpdate** () const
- const bool & **GetShouldDraw** () const

- const bool & **GetIsUI** () const
- const XMUINT2 & **GetLastResolution** () const
- const bool & **GetUseTranslucency** () const
- const XMFLOAT2 & **GetAnchor** () const
- class [CEntity](#) * **GetParent** () const
- const std::string & **GetName** () const
- const std::string **GetDebugInfo** () const
- XMFLOAT3 **GetWorldPosition** ()
Get the position of the component in world space rather than in entity space.
- virtual XMFLOAT4X4 [GetTransform](#) () override
- virtual void [Update](#) (float deltaTime)=0
Updated automatically every single frame.
- virtual void [Draw](#) (struct ID3D11DeviceContext *context, const XMFLOAT4X4 &parentMat, [ConstantBuffer](#) cb, ID3D11Buffer *constantBuffer)=0
Almost the same as [Update\(\)](#) but to be used for drawing only.

Additional Inherited Members

10.15.1 Detailed Description

Fundamental component class of the engine.

Can be extended upon to make new components to add to [CEntity](#).

10.15.2 Member Function Documentation

10.15.2.1 Draw()

```
virtual void CComponent::Draw (
    struct ID3D11DeviceContext * context,
    const XMFLOAT4X4 & parentMat,
    ConstantBuffer cb,
    ID3D11Buffer * constantBuffer ) [pure virtual]
```

Almost the same as [Update\(\)](#) but to be used for drawing only.

Implemented in [CSpriteComponent](#), [CTextRenderComponent](#), [WeaponInterface](#), [Weapon](#), [CAudioEmitterComponent](#), [CParticleEmitter](#), [CRigidBodyComponent](#), and [CCameraComponent](#).

10.15.2.2 GetTransform()

```
XMFLOAT4X4 CComponent::GetTransform ( ) [override], [virtual]
```

Reimplemented from [CTransform](#).

10.15.2.3 SetAnchor()

```
void CComponent::SetAnchor (
    const XMFLOAT2 & newAnchor ) [inline]
```

Sets the region of the screen a UI element will be "anchored" to.

{0,0} - top left, {1,1} - bottom right. Used for making UI elements stick to the edge of the screen when the window is resized.

10.15.2.4 SetUseTranslucency()

```
void CComponent::SetUseTranslucency (
    const bool & newTranslucency ) [virtual]
```

Sets if this component will/can draw translucent pixels.

THIS FUNCTION IS COSTLY - do NOT micro-manage! Use this function once per component and leave it. Will either put the component into the opaque unsorted draw or translucent sorted draw. Translucent components have a much higher overhead than opaque components.

Reimplemented in [CSpriteComponent](#).

10.15.2.5 Update()

```
virtual void CComponent::Update (
    float deltaTime ) [pure virtual]
```

Updated automatically every single frame.

Implemented in [CAudioEmitterComponent](#), [CParticleEmitter](#), [CRigidBodyComponent](#), [Crossbow](#), [CAnimationSpriteComponent](#), [CCameraComponent](#), [CSpriteComponent](#), [CTextRenderComponent](#), [WeaponInterface](#), [Weapon](#), and [Pickup](#).

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

- [CComponent.h](#)
- [CComponent.cpp](#)

10.16 CellData Struct Reference

Public Attributes

- int **id**
- CellType **type**

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

- [CWorld_Edit.h](#)

10.17 CEmitter Class Reference

Public Attributes

- [Vector3](#) **position**
- float **range** = 1000
- [CAudio](#) * **audio**
- EMITTERTYPE **type**

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

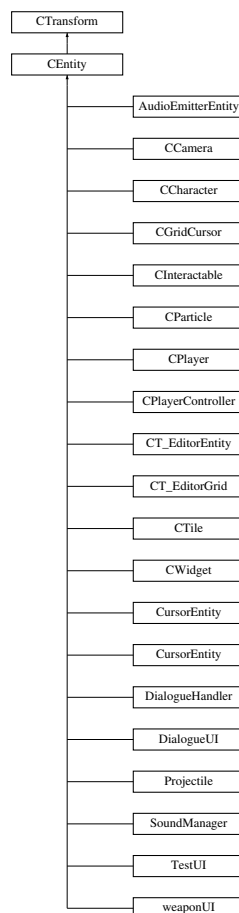
- [CEmitter.h](#)

10.18 CEntity Class Reference

Fundamental class of the engine with a world transform and ability to have components.

```
#include <CEntity.h>
```

Inheritance diagram for CEntity:



Public Member Functions

- void **SetShouldUpdate** (const bool &newShouldUpdate)
Sets if this entity will be automatically updated via the [Update\(\)](#).
- void **SetShouldMove** (const bool &newShouldMove)
Sets whether this entity will move for collision detection.
- void **SetVisible** (const bool &newVisibility)
Sets if this entity and all it's components will be rendered.
- void **SetIsUI** (const bool &newUI)
Sets whether the engine will treat this as UI in the update loop.
- const bool & **GetShouldUpdate** () const
- const bool & **GetShouldMove** () const
- const bool & **GetVisible** () const
- const bool & **GetIsUI** () const
- const std::vector< [CComponent](#) * > & **GetAllComponents** () const
- virtual void **Update** (float deltaTime)=0
Updated automatically every single frame.
- template<class T >
T * **AddComponent** (const std::string &componentName)
- template<class T >
T * **GetComponentOfType** ()
- template<class T >
std::vector< T * > **GetAllComponentsOfType** ()
- void **RemoveComponent** ([CComponent](#) *reference)
Removes the specified component.
- virtual void **HasCollided** ([CollisionComponent](#) *collidedObject)

Public Attributes

- [CollisionComponent](#) * **colComponent** = nullptr

Additional Inherited Members

10.18.1 Detailed Description

Fundamental class of the engine with a world transform and ability to have components.

Use for all gameplay things in the world.

10.18.2 Member Function Documentation

10.18.2.1 HasCollided()

```
virtual void CEntity::HasCollided (
    CollisionComponent * collidedObject ) [inline], [virtual]
```

Reimplemented in [CInteractable](#).

10.18.2.2 SetIsUI()

```
void CEntity::SetIsUI (
    const bool & newUI ) [inline]
```

Sets whether the engine will treat this as UI in the update loop.

I.e. will still be updated when game is paused.

10.18.2.3 Update()

```
virtual void CEntity::Update (
    float deltaTime ) [pure virtual]
```

Updated automatically every single frame.

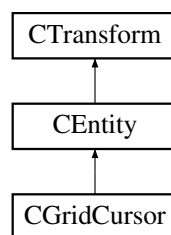
Implemented in [CParticle](#), [CCamera](#), [CCharacter](#), [HomingProjectile](#), [CGridCursor](#), [CTile](#), [CWidget_Button](#), [CWidget_Canvas](#), [CWidget_Image](#), [CWidget_Text](#), [CT_EditorEntity](#), [CT_EditorEntity_WeaponHolder](#), [CT_EditorEntity_Waypoint](#), [CT_EditorEntity_Enemy](#), [CT_EditorEntity_PlayerStart](#), [CT_EditorGrid](#), [CursorEntity](#), [AlarmEnemy](#), [CAIController](#), [DogEnemy](#), [GruntEnemy](#), [AudioEmitterEntity](#), [CInteractable](#), [CPlayer](#), [CursorEntity](#), [DialogueUI](#), [PlayerCharacter](#), [PlayerController](#), [TestUI](#), [PauseMenu](#), [Projectile](#), [SettingsMenu](#), and [weaponUI](#).

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

- [CEntity.h](#)
- [CEntity.cpp](#)

10.19 CGridCursor Class Reference

Inheritance diagram for CGridCursor:



Public Member Functions

- virtual void [Update](#) (float deltaTime) override
Updated automatically every single frame.
- void **UpdateSize** (int X, int Y)

Public Attributes

- class [CSpriteComponent](#) * **activeCellSprite** = nullptr
- [Vector3](#) **Offset**
- [Vector3](#) **Offset_Start**
- [Vector3](#) **Offset_End**
- bool **screenMoved**
- bool **cellInspectingEntity**
- bool **cellSelected**
- [Vector3](#) **selectedCell_1**
- bool **wasMouseReleased**
- class [CCameraComponent](#) * **camera**

Additional Inherited Members

10.19.1 Member Function Documentation

10.19.1.1 Update()

```
void CGridCursor::Update (
    float deltaTime ) [override], [virtual]
```

Updated automatically every single frame.

Implements [CEntity](#).

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

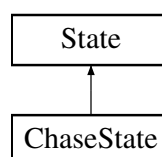
- CGridCursor.h
- CGridCursor.cpp

10.20 ChaseState Class Reference

[State](#) for when the AI is chasing the player.

```
#include <State.h>
```

Inheritance diagram for ChaseState:



Public Member Functions

- void [Enter](#) ([CAIController](#) *controller) override
- void [Update](#) ([CAIController](#) *controller, float deltaTime) override
- void [Exit](#) ([CAIController](#) *controller) override

Static Public Member Functions

- static [State](#) & [getInstance](#) ()

10.20.1 Detailed Description

[State](#) for when the AI is chasing the player.

10.20.2 Member Function Documentation

10.20.2.1 Enter()

```
void ChaseState::Enter (  
    CAIController * controller ) [override], [virtual]
```

Reimplemented from [State](#).

10.20.2.2 Exit()

```
void ChaseState::Exit (  
    CAIController * controller ) [override], [virtual]
```

Reimplemented from [State](#).

10.20.2.3 Update()

```
void ChaseState::Update (  
    CAIController * controller,  
    float deltaTime ) [override], [virtual]
```

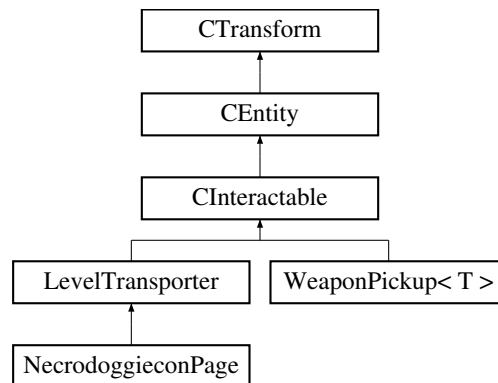
Reimplemented from [State](#).

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

- [State.h](#)
- [State.cpp](#)

10.21 CInteractable Class Reference

Inheritance diagram for CInteractable:



Public Member Functions

- virtual void [Update](#) (float deltaTime) override
Updates the interactables collision component and UI from showing / hiding when within range.
- virtual void [OnInteract](#) ()
Called when a player has interacted with the interactable.
- virtual void [OnEnterOverlap](#) ()
Called when a player is withing range of the interactable.
- virtual void [OnLeaveOverlap](#) ()
Called when a player leaves the range of the interactable.
- virtual void [HasCollided](#) (CollisionComponent *collidedObject) override
Called when a player is colliding with the trigger for the interactable.
- void [SetTexture](#) (std::string path)
Sets the texture for the interactable.
- void [SetTextureWIC](#) (std::string path)
Sets the texture for the interactable.
- void [SetInteractRange](#) (const float value)
Sets the interact range for the interactable.

Protected Member Functions

- void [DrawUI](#) ()
Draws the UI to indicate which key to press to interact with the interactable.
- CollisionComponent * [GetLastCollidedObject](#) ()
Returns the last collided object of the interactable.
- CSpriteComponent * [GetSprite](#) ()
Returns the sprite of the interactable.

Additional Inherited Members

10.21.1 Member Function Documentation

10.21.1.1 GetLastCollidedObject()

```
CollisionComponent * CInteractable::GetLastCollidedObject ( ) [protected]
```

Returns the last collided object of the interactable.

Returns

the collision component pointer of the last collided object.

10.21.1.2 GetSprite()

```
CSpriteComponent * CInteractable::GetSprite ( ) [protected]
```

Returns the sprite of the interactable.

Returns

the sprite of the interactable.

10.21.1.3 HasCollided()

```
void CInteractable::HasCollided (
    CollisionComponent * collidedObject ) [override], [virtual]
```

Called when a player is colliding with the trigger for the interactable.

Parameters

<i>collidedObject</i>	the other object we are colliding with.
-----------------------	---

Reimplemented from [CEntity](#).

10.21.1.4 OnInteract()

```
void CInteractable::OnInteract ( ) [virtual]
```

Called when a player has interacted with the interactable.

Reimplemented in [LevelTransporter](#), [NecrodoggieconPage](#), and [WeaponPickup< T >](#).

10.21.1.5 SetInteractRange()

```
void CInteractable::SetInteractRange (
    const float value )
```

Sets the interact range for the interactable.

Parameters

<i>value</i>	the interact range for the interactable.
--------------	--

10.21.1.6 SetTexture()

```
void CInteractable::SetTexture (
    std::string path )
```

Sets the texture for the interactable.

Parameters

<i>path</i>	the path to the texture used for the interactable.
-------------	--

10.21.1.7 SetTextureWIC()

```
void CInteractable::SetTextureWIC (
    std::string path )
```

Sets the texture for the interactable.

Parameters

<i>path</i>	the path to the texture used for the interactable.
-------------	--

10.21.1.8 Update()

```
void CInteractable::Update (
    float deltaTime ) [override], [virtual]
```

Updates the interactables collision component and UI from showing / hiding when within range.

Parameters

<i>deltaTime</i>	
------------------	--

Implements [CEntity](#).

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

- [CInteractable.h](#)
- CInteractable.cpp

10.22 CMaterial Struct Reference

Holds the directx stuff for uploading sprite specific data to the shader.

```
#include <CMaterial.h>
```

Public Member Functions

- HRESULT **CreateMaterial** (XMUINT2 texSize)
- void **UpdateMaterial** ()

Public Attributes

- [MaterialPropertiesConstantBuffer](#) **material**
- ID3D11Buffer * **materialConstantBuffer** = nullptr
- bool **loaded** = false

10.22.1 Detailed Description

Holds the directx stuff for uploading sprite specific data to the shader.

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

- [CMaterial.h](#)
- CMaterial.cpp

10.23 CMesh Struct Reference

Holds all information about a mesh for use by [CSpriteComponent](#).

```
#include <CMesh.h>
```

Public Member Functions

- HRESULT **LoadMesh** ()

Public Attributes

- ID3D11Buffer * **vertexBuffer**
- ID3D11Buffer * **indexBuffer**
- bool **loaded** = false

10.23.1 Detailed Description

Holds all information about a mesh for use by [CSpriteComponent](#).

Right now only stores a hardcoded quad - might need extending in future for new shapes.

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

- [CMesh.h](#)
- [CMesh.cpp](#)

10.24 CollisionComponent Class Reference

Public Member Functions

- **CollisionComponent** (std::string setName, [CEntity](#) *parent)
- COLLISIONTYPE **GetCollisionType** ()
- float **GetRadius** ()
- void **SetRadius** (float setRadius)
- void **SetPosition** ([Vector3](#) setPosition)
- [Vector3](#) **GetPosition** ()
- std::string **GetName** ()
- float **GetWidth** ()
- float **GetHeight** ()
- bool **Intersects** ([CollisionComponent](#) *circle, [CollisionComponent](#) *box)
- void **SetCollider** (float setRadius)
- void **SetCollider** (float setHeight, float setWidth)
- bool **IsColliding** ([CollisionComponent](#) *collidingObject)
- float **DistanceBetweenPoints** ([Vector3](#) &point1, [Vector3](#) &point2)
- [CEntity](#) * **GetParent** ()
- void **Resolve** ([CollisionComponent](#) *other)
Resolves collisions between two collider's.
- void **SetTrigger** (const bool value)
- bool **GetTrigger** ()

10.24.1 Member Function Documentation

10.24.1.1 Resolve()

```
void CollisionComponent::Resolve (
    CollisionComponent * other )
```

Resolves collisions between two collider's.

Parameters

other	
-------	--

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

- CollisionComponent.h
- CollisionComponent.cpp

10.25 ConstantBuffer Struct Reference

Public Attributes

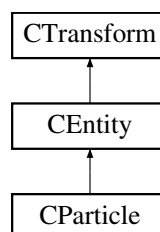
- XMATRIX **mWorld**
- XMATRIX **mView**
- XMATRIX **mProjection**
- XMFLOAT4 **vOutputColor**

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

- structures.h

10.26 CParticle Class Reference

Inheritance diagram for CParticle:



Public Member Functions

- virtual void **Update** (float deltaTime)
Updates the particles lifetime and velocity.
- void **Draw** (ID3D11DeviceContext *context, const XMATRIX &parentMat, **ConstantBuffer** cb, ID3D11Buffer *constantBuffer)
Draws the particle.
- void **SetLifetime** (const float life)
Sets the lifetime of the particle.
- float **GetLifetime** ()
Returns the lifetime of the particle.
- void **SetVelocity** (const float velo)

- Sets the velocity of the particle.*
 - float [GetVelocity](#) ()*Returns the velocity of the particle.*
- void [SetDirection](#) (const [Vector3](#) dir)*Sets the direction of the particle.*
- [Vector3](#) [GetDirection](#) ()*Returns the direction of the particle.*
- [CSpriteComponent](#) * [getSpriteComponent](#) ()*Returns the sprite component of the particle.*

Additional Inherited Members

10.26.1 Member Function Documentation

10.26.1.1 Draw()

```
void CParticle::Draw (
    ID3D11DeviceContext * context,
    const XMFLLOAT4X4 & parentMat,
    ConstantBuffer cb,
    ID3D11Buffer * constantBuffer )
```

Draws the particle.

Parameters

<i>context</i>	
<i>parentMat</i>	
<i>cb</i>	
<i>constantBuffer</i>	

10.26.1.2 GetDirection()

```
Vector3 CParticle::GetDirection ( )
```

Returns the direction of the particle.

Returns

the direction of the particle.

10.26.1.3 GetLifetime()

```
float CParticle::GetLifetime ( )
```

Returns the lifetime of the particle.

Returns

the lifetime of the particle.

10.26.1.4 getSpriteComponent()

```
CSpriteComponent * CParticle::getSpriteComponent ( )
```

Returns the sprite component of the particle.

Returns

the sprite component of the particle.

10.26.1.5 GetVelocity()

```
float CParticle::GetVelocity ( )
```

Returns the velocity of the particle.

Returns

the velocity of the particle.

10.26.1.6 SetDirection()

```
void CParticle::SetDirection (
    const Vector3 dir )
```

Sets the direction of the particle.

Parameters

<i>dir</i>	the direction of the particle.
------------	--------------------------------

10.26.1.7 SetLifetime()

```
void CParticle::SetLifetime (
    const float life )
```

Sets the lifetime of the particle.

Parameters

<i>life</i>	the lifetime of the particle
-------------	------------------------------

10.26.1.8 SetVelocity()

```
void CParticle::SetVelocity (
    const float velo )
```

Sets the velocity of the particle.

Parameters

<i>velo</i>	the velocity of the particle.
-------------	-------------------------------

10.26.1.9 Update()

```
void CParticle::Update (
    float deltaTime ) [virtual]
```

Updates the particles lifetime and velocity.

Parameters

<i>deltaTime</i>	
------------------	--

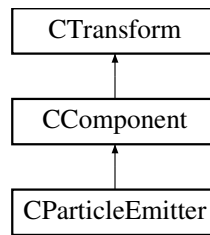
Implements [CEntity](#).

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

- CParticle.h
- [CParticle.cpp](#)

10.27 CParticleEmitter Class Reference

Inheritance diagram for CParticleEmitter:



Public Member Functions

- void [SetTexture](#) (const std::string &path)
Sets the texture for the particles emitted.
- void [SetSize](#) (const int size)
Sets the ammount of particles in the emitter.
- void [UseRandomDirection](#) (bool toggle, const [Vector3](#) min, const [Vector3](#) max)
Toggles use of random direction.
- void [UseRandomVelocity](#) (bool toggle, const float min, const float max)
Toggles use of random velocity.
- void [UseRandomLifetime](#) (bool toggle, const float min, const float max)
Toggles use of random lifetime.
- void [SetDirection](#) (const [Vector3](#) dir)
Sets the overall particle direction.
- [Vector3](#) [GetDirection](#) ()
Returns the overall particle direction.
- void [SetVelocity](#) (const float velo)
Sets the overall particle velocity.
- float [GetVelocity](#) ()
Returns the overall particle velocity.
- void [SetLifetime](#) (const float life)
Sets the overall particles lifetime.
- float [GetLifetime](#) ()
Returns the overall particles lifetime.
- void **Start** ()
Starts the emitter that emits particles.
- void **Stop** ()
Stops the emitter from emitting particles.
- virtual void [Update](#) (float deltaTime)
Updates the particles in the emitter (i.e.
- virtual void [Draw](#) (struct ID3D11DeviceContext *context, const XMFLOAT4X4 &parentMat, [ConstantBuffer](#) cb, ID3D11Buffer *constantBuffer)
Draws the particles in relation to the emitters transform.

Additional Inherited Members

10.27.1 Member Function Documentation

10.27.1.1 Draw()

```
void CParticleEmitter::Draw (
    struct ID3D11DeviceContext * context,
    const XMFLOAT4X4 & parentMat,
    ConstantBuffer cb,
    ID3D11Buffer * constantBuffer ) [virtual]
```

Draws the particles in relation to the emitters transform.

Parameters

<i>context</i>	
<i>parentMat</i>	
<i>cb</i>	
<i>constantBuffer</i>	

Implements [CComponent](#).

10.27.1.2 GetDirection()

```
Vector3 CParticleEmitter::GetDirection ( )
```

Returns the overall particle direction.

Returns

the direction of all particles.

10.27.1.3 GetLifetime()

```
float CParticleEmitter::GetLifetime ( )
```

Returns the overall particles lifetime.

Returns

lifetime of particle

10.27.1.4 GetVelocity()

```
float CParticleEmitter::GetVelocity ( )
```

Returns the overall particle velocity.

Returns

velocity of particle

10.27.1.5 SetDirection()

```
void CParticleEmitter::SetDirection (
    const Vector3 dir )
```

Sets the overall particle direction.

Parameters

<i>dir</i>	the direction of all particles.
------------	---------------------------------

10.27.1.6 SetLifetime()

```
void CParticleEmitter::SetLifetime (
    const float life )
```

Sets the overall particles lifetime.

Parameters

<i>life</i>	the lifetime of all particles.
-------------	--------------------------------

10.27.1.7 SetSize()

```
void CParticleEmitter::SetSize (
    const int size )
```

Sets the ammount of particles in the emitter.

Parameters

<i>size</i>	the ammount of particles used in the emitter.
-------------	---

10.27.1.8 SetTexture()

```
void CParticleEmitter::SetTexture (
    const std::string & path )
```

Sets the texture for the particles emitted.

Parameters

<i>path</i>	the path to the texture for the particles.
-------------	--

10.27.1.9 SetVelocity()

```
void CParticleEmitter::SetVelocity (
    const float velo )
```

Sets the overall particle velocity.

Parameters

<i>velo</i>	the velocity of all particles.
-------------	--------------------------------

10.27.1.10 Update()

```
void CParticleEmitter::Update (
    float deltaTime ) [virtual]
```

Updates the particles in the emitter (i.e.

Movement and lifetime of each particle).

Parameters

<i>deltaTime</i>	
------------------	--

Implements [CComponent](#).

10.27.1.11 UseRandomDirection()

```
void CParticleEmitter::UseRandomDirection (
    bool toggle,
```

```
const Vector3 min,  
const Vector3 max )
```

Toggles use of random direction.

Parameters

<i>toggle</i>	- boolean value toggling random usage.
<i>min</i>	- minimum random value.
<i>max</i>	- maximum random value.

10.27.1.12 UseRandomLifetime()

```
void CParticleEmitter::UseRandomLifetime (  
    bool toggle,  
    const float min,  
    const float max )
```

Toggles use of random lifetime.

Parameters

<i>toggle</i>	- boolean value toggling random usage.
<i>min</i>	- minimum random value.
<i>max</i>	- maximum random value.

10.27.1.13 UseRandomVelocity()

```
void CParticleEmitter::UseRandomVelocity (  
    bool toggle,  
    const float min,  
    const float max )
```

Toggles use of random velocity.

Parameters

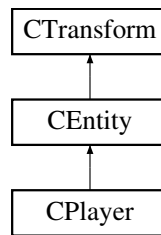
<i>toggle</i>	- boolean value toggling random usage.
<i>min</i>	- minimum random value.
<i>max</i>	- maximum random value.

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

- [CParticleEmitter.h](#)
- [CParticleEmitter.cpp](#)

10.28 CPlayer Class Reference

Inheritance diagram for CPlayer:



Public Member Functions

- virtual void [Update](#) (float deltaTime) override
Updated automatically every single frame.

Additional Inherited Members

10.28.1 Member Function Documentation

10.28.1.1 Update()

```
void CPlayer::Update (  
    float deltaTime ) [override], [virtual]
```

Updated automatically every single frame.

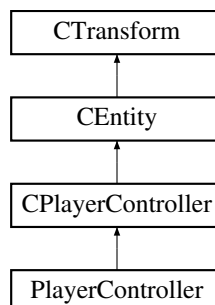
Implements [CEntity](#).

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

- CPlayer.h
- CPlayer.cpp

10.29 CPlayerController Class Reference

Inheritance diagram for CPlayerController:



Public Member Functions

- void **Possess** ([CCharacter](#) *characterToPossess)
Function used to possess a new Character Will Unpossess the Controllers current Character and then set the current Character to the Character that was passed in.
- void **Unpossess** ()
Function used to unpossess a Character Will remove all data associated with the current Character from the Controller.

Protected Member Functions

- [CCharacter](#) * **GetCharacter** ()
- bool **HasCharacter** ()
- virtual void **HandleInput** (float deltaTime)
Virtual function used to handle the input that the controller receives.
- virtual void **OnPossess** ()
- virtual void **OnUnpossess** ()

Additional Inherited Members

10.29.1 Member Function Documentation

10.29.1.1 HandleInput()

```
void CPlayerController::HandleInput (
    float deltaTime ) [protected], [virtual]
```

Virtual function used to handle the input that the controller receives.

Reimplemented in [PlayerController](#).

10.29.1.2 OnPossess()

```
virtual void CPlayerController::OnPossess ( ) [inline], [protected], [virtual]
```

Reimplemented in [PlayerController](#).

10.29.1.3 OnUnpossess()

```
virtual void CPlayerController::OnUnpossess ( ) [inline], [protected], [virtual]
```

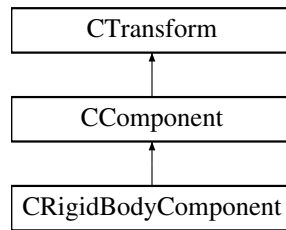
Reimplemented in [PlayerController](#).

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

- CPlayerController.h
- [CPlayerController.cpp](#)

10.30 CRigidBodyComponent Class Reference

Inheritance diagram for CRigidBodyComponent:



Public Member Functions

- virtual void [Update](#) (float deltaTime)
Updates the integration for the rigid body system.
- virtual void [Draw](#) (struct ID3D11DeviceContext *context, const XMFLOAT4X4 &parentMat, [ConstantBuffer](#) cb, ID3D11Buffer *constantBuffer)
Almost the same as [Update\(\)](#) but to be used for drawing only.
- void [SetVelocity](#) (const [Vector3](#) &velo)
Sets the velocity of the rigidbody.
- [Vector3](#) & [GetVelocity](#) ()
Returns the current velocity of the rigidbody.
- void [SetAcceleration](#) (const [Vector3](#) &accel)
Sets the acceleration of the rigidbody.
- [Vector3](#) & [GetAcceleration](#) ()
Returns the current acceleration of the rigidbody.

Additional Inherited Members

10.30.1 Member Function Documentation

10.30.1.1 Draw()

```

virtual void CRigidBodyComponent::Draw (
    struct ID3D11DeviceContext * context,
    const XMFLOAT4X4 & parentMat,
    ConstantBuffer cb,
    ID3D11Buffer * constantBuffer ) [inline], [virtual]
  
```

Almost the same as [Update\(\)](#) but to be used for drawing only.

Implements [CComponent](#).

10.30.1.2 GetAcceleration()

```
Vector3 & CRigidBodyComponent::GetAcceleration ( )
```

Returns the current acceleration of the rigidbody.

Returns

10.30.1.3 GetVelocity()

```
Vector3 & CRigidBodyComponent::GetVelocity ( )
```

Returns the current velocity of the rigidbody.

Returns

10.30.1.4 SetAcceleration()

```
void CRigidBodyComponent::SetAcceleration (
    const Vector3 & accel )
```

Sets the acceleration of the rigidbody.

Parameters

<i>accel</i>	
--------------	--

10.30.1.5 SetVelocity()

```
void CRigidBodyComponent::SetVelocity (
    const Vector3 & velo )
```

Sets the velocity of the rigidbody.

Parameters

<i>velo</i>	
-------------	--

10.30.1.6 Update()

```
void CRigidBodyComponent::Update (
    float deltaTime ) [virtual]
```

Updates the integration for the rigid body system.

Parameters

<i>deltaTime</i>	
------------------	--

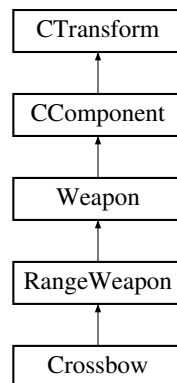
Implements [CComponent](#).

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

- CRigidBodyComponent.h
- [CRigidBodyComponent.cpp](#)

10.31 Crossbow Class Reference

Inheritance diagram for Crossbow:



Public Member Functions

- virtual void [Update](#) (float deltaTime)
Update function for Cooldown of weapons.

Additional Inherited Members

10.31.1 Member Function Documentation

10.31.1.1 Update()

```
void Crossbow::Update (  
    float deltaTime ) [virtual]
```

Update function for Cooldown of weapons.

Parameters

<i>deltaTime</i>	
------------------	--

Reimplemented from [Weapon](#).

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

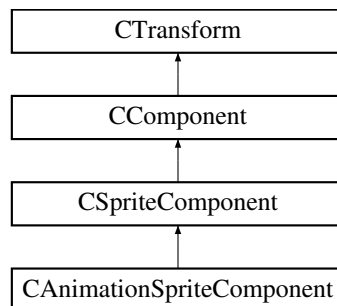
- [Crossbow.h](#)
- [Crossbow.cpp](#)

10.32 CSpriteComponent Class Reference

A component for loading and displaying a 2D texture in world space as part of [CEntity](#).

```
#include <CSpriteComponent.h>
```

Inheritance diagram for CSpriteComponent:



Public Member Functions

- virtual void [SetRenderRect](#) (const XMUINT2 &newSize)
Used to resize the portion of the texture you want to display on the sprite in pixels.
- void [SetTextureOffset](#) (const XMFLOAT2 &newOffset)
The offset in pixels of where the sprite should start rendering in the texture.
- virtual void [SetSpriteSize](#) (const XMUINT2 &newSize)
The size of the ingame sprite in pixels.
- void [SetTint](#) (const XMFLOAT4 &newTint)
Set the color tint of the sprite in RGBA.
- virtual void [SetUseTranslucency](#) (const bool &newTranslucency) override
Sets if this component will/can draw translucent pixels.
- HRESULT [LoadTexture](#) (const std::string &filePath)
Loads the texture from a file.
- HRESULT [LoadTextureWIC](#) (const std::string &filePath)
Loads the texture from a file.
- const XMUINT2 & [GetRenderRect](#) () const
- const XMFLOAT2 & [GetTextureOffset](#) () const
- const XMUINT2 & [GetSpriteSize](#) () const
- const XMFLOAT4 & [GetTint](#) () const
- const XMUINT2 & [GetTextureSize](#) () const
- virtual XMFLOAT4X4 [GetTransform](#) () override
- virtual void [Update](#) (float deltaTime) override
Updated automatically every single frame.
- virtual void [Draw](#) (ID3D11DeviceContext *context, const XMFLOAT4X4 &parentMat, [ConstantBuffer](#) cb, ID3D11Buffer *constantBuffer) override
Almost the same as [Update\(\)](#) but to be used for drawing only.

Additional Inherited Members

10.32.1 Detailed Description

A component for loading and displaying a 2D texture in world space as part of [CEntity](#).

10.32.2 Member Function Documentation

10.32.2.1 Draw()

```
void CSpriteComponent::Draw (
    ID3D11DeviceContext * context,
    const XMFLOAT4X4 & parentMat,
    ConstantBuffer cb,
    ID3D11Buffer * constantBuffer ) [override], [virtual]
```

Almost the same as [Update\(\)](#) but to be used for drawing only.

Implements [CComponent](#).

10.32.2.2 GetTransform()

```
XMFLOAT4X4 CSpriteComponent::GetTransform ( ) [override], [virtual]
```

Reimplemented from [CComponent](#).

10.32.2.3 LoadTexture()

```
HRESULT CSpriteComponent::LoadTexture (
    const std::string & filePath )
```

Loads the texture from a file.

MUST use the .dds file type.

10.32.2.4 LoadTextureWIC()

```
HRESULT CSpriteComponent::LoadTextureWIC (
    const std::string & filePath )
```

Loads the texture from a file.

MUST use BMP, JPEG, PNG, TIFF, GIF, or HD Photo file types.

10.32.2.5 SetRenderRect()

```
void CSpriteComponent::SetRenderRect (
    const XMUINT2 & newSize ) [virtual]
```

Used to resize the portion of the texture you want to display on the sprite in pixels.

Use to set the size of a selection of a sprite sheet.

10.32.2.6 SetSpriteSize()

```
virtual void CSpriteComponent::SetSpriteSize (
    const XMUINT2 & newSize ) [inline], [virtual]
```

The size of the ingame sprite in pixels.

Set automatically on texture load.

10.32.2.7 SetTextureOffset()

```
void CSpriteComponent::SetTextureOffset (
    const XMFLOAT2 & newOffset )
```

The offset in pixels of where the sprite should start rendering in the texture.

Use this for selecting a section of a sprite sheet. By default set to 0,0.

10.32.2.8 SetUseTranslucency()

```
void CSpriteComponent::SetUseTranslucency (
    const bool & newTranslucency ) [override], [virtual]
```

Sets if this component will/can draw translucent pixels.

THIS FUNCTION IS COSTLY - do NOT micro-manage! Use this function once per component and leave it. Will either put the component into the opaque unsorted draw or translucent sorted draw. Translucent components have a much higher overhead than opaque components.

Reimplemented from [CComponent](#).

10.32.2.9 Update()

```
void CSpriteComponent::Update (
    float deltaTime ) [override], [virtual]
```

Updated automatically every single frame.

Implements [CComponent](#).

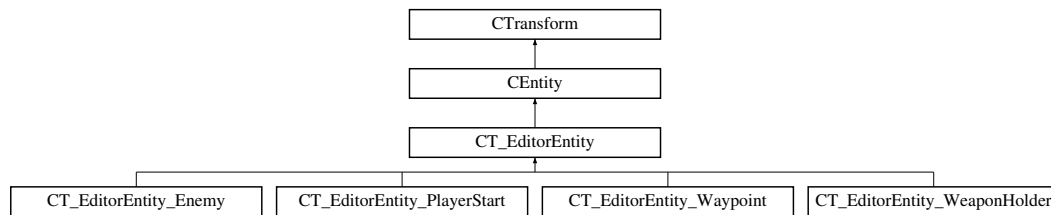
Reimplemented in [CAAnimationSpriteComponent](#).

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

- [CSpriteComponent.h](#)
- [CSpriteComponent.cpp](#)

10.33 CT_EditorEntity Class Reference

Inheritance diagram for CT_EditorEntity:



Public Member Functions

- virtual void [Update](#) (float deltaTime) override
Updated automatically every single frame.
- virtual void **InitialiseEntity** (int SlotID)
- EditorEntityType **GetType** ()
- int **GetSlot** ()

Public Attributes

- class [CSpriteComponent](#) * **sprite** = nullptr

Protected Attributes

- int **entitySlotID**
- EditorEntityType **inspectType**

10.33.1 Member Function Documentation

10.33.1.1 Update()

```
void CT_EditorEntity::Update (
    float deltaTime ) [override], [virtual]
```

Updated automatically every single frame.

Implements [CEntity](#).

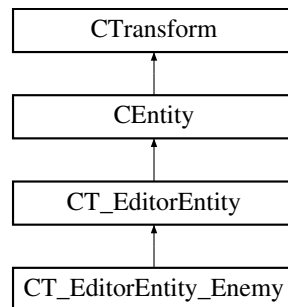
Reimplemented in [CT_EditorEntity_WeaponHolder](#), [CT_EditorEntity_Waypoint](#), [CT_EditorEntity_Enemy](#), and [CT_EditorEntity_PlayerStart](#).

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

- CT_EditorEntity.h
- CT_EditorEntity.cpp

10.34 CT_EditorEntity_Enemy Class Reference

Inheritance diagram for CT_EditorEntity_Enemy:



Public Member Functions

- float **GetHealth** ()
- float **GetSpeed** ()
- float **GetMass** ()
- float **GetRange** ()
- float **GetViewAngle** ()
- float **GetRotationSpeed** ()
- float **GetMaxSearchTime** ()
- bool **GetIsBoss** ()
- void **SetHealth** (float newHealth)
- void **SetSpeed** (float newSpeed)
- void **SetMass** (float newMass)
- void **SetRange** (float newRange)
- void **SetViewAngle** (float newViewAngle)
- void **SetRotationSpeed** (float newRotationSpeed)
- void **SetMaxSearchTime** (float newMaxSearchTime)
- void **SetIsBoss** (bool newIsBoss)
- char * **GetWeaponName** ()
- int **GetAssignedWeapon** ()
- void **AssignWeapon** (char *WeaponID, int Index)
- virtual void **Update** (float deltaTime) override
Updated automatically every single frame.
- virtual void **InitialiseEntity** (int SlotID)
- void **ToggleWaypoints** (bool Display)
- **CT_EditorEntity_Waypoint** * **AddWaypoint** (**Vector2** Position)
- void **RemoveWaypoint** (int Index)

Public Attributes

- std::vector< **CT_EditorEntity_Waypoint** * > **Waypoints**

Protected Attributes

- bool **displayWaypoints** = false
- char * **current_item** = (char*)"Dagger"
- int **itemIndex** = 0
- float **health** = 2.0f
- float **speed** = 100.0f
- float **mass** = 10.0f
- float **range** = 200.0f
- float **viewAngle** = 90.0f
- float **rotationSpeed** = 0.01f
- float **maxSearchTime** = 5.0f
- bool **isBoss** = false

10.34.1 Member Function Documentation

10.34.1.1 InitialiseEntity()

```
void CT_EditorEntity_Enemy::InitialiseEntity (
    int SlotID ) [virtual]
```

Reimplemented from [CT_EditorEntity](#).

10.34.1.2 Update()

```
void CT_EditorEntity_Enemy::Update (
    float deltaTime ) [override], [virtual]
```

Updated automatically every single frame.

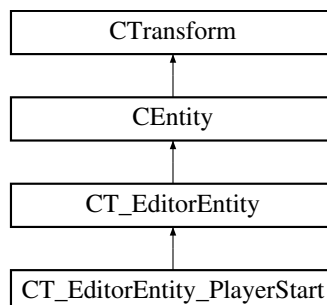
Reimplemented from [CT_EditorEntity](#).

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

- CT_EditorEntity.h
- CT_EditorEntity.cpp

10.35 CT_EditorEntity_PlayerStart Class Reference

Inheritance diagram for CT_EditorEntity_PlayerStart:



Public Member Functions

- virtual void [Update](#) (float deltaTime) override
Updated automatically every single frame.

Additional Inherited Members

10.35.1 Member Function Documentation

10.35.1.1 Update()

```
void CT_EditorEntity_PlayerStart::Update (
    float deltaTime ) [override], [virtual]
```

Updated automatically every single frame.

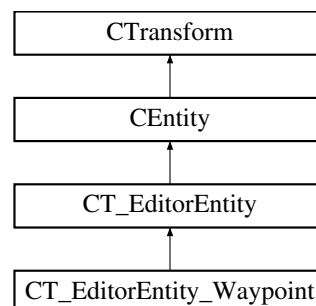
Reimplemented from [CT_EditorEntity](#).

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

- CT_EditorEntity.h
- CT_EditorEntity.cpp

10.36 CT_EditorEntity_Waypoint Class Reference

Inheritance diagram for CT_EditorEntity_Waypoint:



Public Member Functions

- [Vector2](#) [GetGridPos](#) ()
- virtual void [Update](#) (float deltaTime) override
Updated automatically every single frame.
- virtual void [InitialiseEntity](#) (int SlotID)

Public Attributes

- int **waypointOrder**
- [Vector2](#) **gridPos**

Additional Inherited Members

10.36.1 Member Function Documentation

10.36.1.1 InitialiseEntity()

```
void CT_EditorEntity_Waypoint::InitialiseEntity (
    int SlotID ) [virtual]
```

Reimplemented from [CT_EditorEntity](#).

10.36.1.2 Update()

```
void CT_EditorEntity_Waypoint::Update (
    float deltaTime ) [override], [virtual]
```

Updated automatically every single frame.

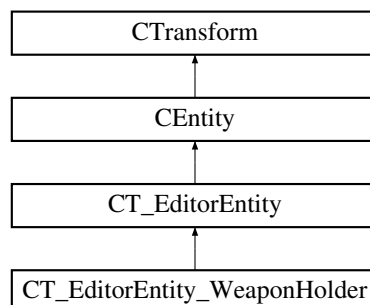
Reimplemented from [CT_EditorEntity](#).

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

- CT_EditorEntity.h
- CT_EditorEntity.cpp

10.37 CT_EditorEntity_WeaponHolder Class Reference

Inheritance diagram for CT_EditorEntity_WeaponHolder:



Public Member Functions

- char * **GetWeaponName** ()
- int **GetAssignedWeapon** ()
- void **AssignWeapon** (char *WeaponID, int Index)
- virtual void **Update** (float deltaTime) override
Updated automatically every single frame.
- virtual void **InitialiseEntity** (int SlotID)

Protected Attributes

- char * **current_item** = (char*)"Dagger"
- int **itemSlot** = 0
- [CSpriteComponent](#) * **weaponSprite**

Additional Inherited Members

10.37.1 Member Function Documentation

10.37.1.1 InitialiseEntity()

```
void CT_EditorEntity_WeaponHolder::InitialiseEntity (  
    int SlotID ) [virtual]
```

Reimplemented from [CT_EditorEntity](#).

10.37.1.2 Update()

```
void CT_EditorEntity_WeaponHolder::Update (  
    float deltaTime ) [override], [virtual]
```

Updated automatically every single frame.

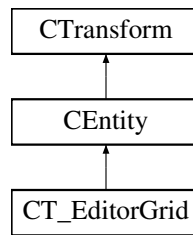
Reimplemented from [CT_EditorEntity](#).

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

- CT_EditorEntity.h
- CT_EditorEntity.cpp

10.38 CT_EditorGrid Class Reference

Inheritance diagram for CT_EditorGrid:



Public Member Functions

- virtual void [Update](#) (float deltaTime) override
Updated automatically every single frame.
- void **SetupGrid** ()
- void **SetupGrid** (class [CCameraComponent](#) *cam)

Public Attributes

- class [CGridCursor](#) * **cursorEntity**

Protected Attributes

- class [CSpriteComponent](#) * **gridSprite** = nullptr

10.38.1 Member Function Documentation

10.38.1.1 Update()

```
void CT_EditorGrid::Update (  
    float deltaTime ) [override], [virtual]
```

Updated automatically every single frame.

Implements [CEntity](#).

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

- CT_EditorGrid.h
- CT_EditorGrid.cpp

10.39 CT_EditorMain Class Reference

Public Member Functions

- void **Initialise** ()
- void **RenderWindows** ()

Public Attributes

- class [CT_EditorGrid](#) * **grid**
- class [CT_EditorWindows](#) * **editorWindow**

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

- CT_EditorMain.h
- CT_EditorMain.cpp

10.40 CT_EditorWindows Class Reference

Public Member Functions

- void **ClearLog** ()
- void **AddLog** (const char *fmt,...) IM_FMTARGS(2)
- void **LoadWeapons** ()
- void **InitialiseMapSlot** ()
- void **render** ()

Protected Attributes

- const char * **WindowTitle** = "Editor Window"
- [Vector2](#) **WindowScale** = (256.0f, 256.0f)

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

- CT_EditorWindows.h
- CT_EditorWindows.cpp

10.41 CT_PropData Struct Reference

Public Member Functions

- **CT_PropData** (int ID, int Coordinate)

Public Attributes

- int **propID**
- [Vector3](#) **coordinate**

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

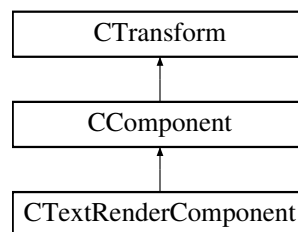
- WorldConstants.h

10.42 CTextRenderComponent Class Reference

A component for rendering text to the screen from a sprite-sheet.

```
#include <CTextRenderComponent.h>
```

Inheritance diagram for CTextRenderComponent:



Public Member Functions

- HRESULT **SetFont** (std::string filePath)
Sets the sprite-sheet for use by the text sprites.
- void **SetText** (std::string newText)
Sets the text to be rendered by the component.
- void **SetReserveCount** (unsigned short newReserveCount)
Sets the minimum amount of sprites to be loaded in memory at any time.
- void **SetJustification** ([TextJustification](#) newJustification)
Sets how the text will justified to the center of the component.
- void **SetCharacterSize** (XMUINT2 newSize)
Sets how big in pixels the characters are from the sprite sheet.
- void **SetCharacterDrawSize** (XMUINT2 newSize)
Set the size of a character when drawn in pixels.
- void **SetSpriteSheetColumnsCount** (unsigned short newColumnsCount)
Set how many columns are in the font sprite sheet.
- const std::string & **GetText** () const
- const unsigned short & **GetReserveCount** () const
- const XMUINT2 & **GetCharacterSize** () const
- const XMUINT2 & **GetCharacterDrawSize** () const
- const unsigned short & **SetSpriteSheetColumnsCount** () const
- virtual void **Update** (float deltaTime) override
Updated automatically every single frame.
- virtual void **Draw** (ID3D11DeviceContext *context, const XMFLOAT4X4 &parentMat, [ConstantBuffer](#) cb, ID3D11Buffer *constantBuffer) override
Almost the same as [Update\(\)](#) but to be used for drawing only.

Additional Inherited Members

10.42.1 Detailed Description

A component for rendering text to the screen from a sprite-sheet.

10.42.2 Member Function Documentation

10.42.2.1 Draw()

```
void CTextRenderComponent::Draw (
    ID3D11DeviceContext * context,
    const XMFLLOAT4X4 & parentMat,
    ConstantBuffer cb,
    ID3D11Buffer * constantBuffer ) [override], [virtual]
```

Almost the same as [Update\(\)](#) but to be used for drawing only.

Implements [CComponent](#).

10.42.2.2 SetCharacterSize()

```
void CTextRenderComponent::SetCharacterSize (
    XMUINT2 newSize )
```

Sets how big in pixels the characters are from the sprite sheet.

Similar to SetRenderRect of [CSpriteComponent](#).

10.42.2.3 SetJustification()

```
void CTextRenderComponent::SetJustification (
    TextJustification newJustification )
```

Sets how the text will justified to the center of the component.

Just look at justification in MS Word.

10.42.2.4 SetReserveCount()

```
void CTextRenderComponent::SetReserveCount (
    unsigned short newReserveCount )
```

Sets the minimum amount of sprites to be loaded in memory at any time.

Lower values will use less memory but will require extra sprites to be created if number of characters to display exceeds the reserve.

10.42.2.5 SetSpriteSheetColumnsCount()

```
void CTextRenderComponent::SetSpriteSheetColumnsCount (
    unsigned short newColumnsCount )
```

Set how many columns are in the font sprite sheet.

If 16 characters across, put 16.

10.42.2.6 Update()

```
void CTextRenderComponent::Update (
    float deltaTime ) [override], [virtual]
```

Updated automatically every single frame.

Implements [CComponent](#).

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

- [CTextRenderComponent.h](#)
- CTextRenderComponent.cpp

10.43 CTexture Struct Reference

Holds all information about a texture for use by [CSpriteComponent](#).

```
#include <CTexture.h>
```

Public Member Functions

- HRESULT **LoadTextureDDS** (std::string filePath)
- HRESULT **LoadTextureWIC** (std::string filename)

Public Attributes

- XMUINT2 **textureSize** = {0,0}
- ID3D11ShaderResourceView * **textureResourceView**
- ID3D11SamplerState * **samplerLinear**
- bool **loaded** = false

10.43.1 Detailed Description

Holds all information about a texture for use by [CSpriteComponent](#).

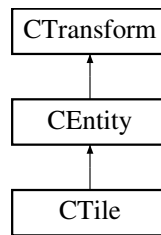
Use load function to populate.

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

- [CTexture.h](#)
- CTexture.cpp

10.44 CTile Class Reference

Inheritance diagram for CTile:



Public Member Functions

- **CTile** (int TileID, [Vector3](#) Position)
- virtual void [Update](#) (float deltaTime) override
Updated automatically every single frame.
- void **ChangeTileID** (CellID TileID)
- void **ChangeTileID** (int ID)
- int **GetTileID** ()
- std::vector< int > **GetConnectedTiles** ()
- void **AddConnectedTile** (int Tile)
- void **SetNavID** (int ID)
- int **GetNavID** ()
- bool **IsWalkable** ()
- void **SetDebugMode** (bool newState)
- void **UpdateDebugRender** ()

Public Attributes

- class [CSpriteComponent](#) * **sprite** = nullptr
- class [CSpriteComponent](#) * **debugSprite** = nullptr

Protected Member Functions

- TileType **GetTileType** ()

Additional Inherited Members

10.44.1 Member Function Documentation

10.44.1.1 Update()

```
void CTile::Update (
    float deltaTime ) [override], [virtual]
```

Updated automatically every single frame.

Implements [CEntity](#).

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

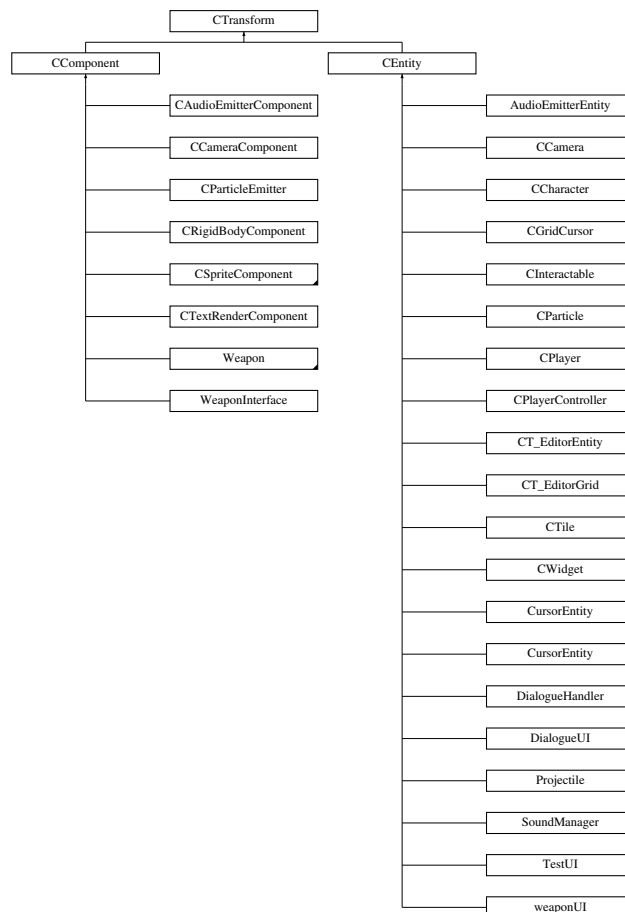
- CTile.h
- CTile.cpp

10.45 CTransform Class Reference

A transform class that contains getters and setters.

```
#include <CTransform.h>
```

Inheritance diagram for CTransform:



Public Member Functions

- void **SetPosition** (const float &x, const float &y, const float &z)
- void **SetScale** (const float &x, const float &y, const float &z)
- void **SetPosition** (const [Vector3](#) &In)
- void **SetScale** (const [Vector3](#) &In)
- void **SetRotation** (const float &Rot)
- const [Vector3](#) & **GetPosition** () const
- const [Vector3](#) & **GetScale** () const
- const float & **GetRotation** () const
- virtual XMFLOAT4X4 **GetTransform** ()

Protected Attributes

- bool **updateTransform** = true
- XMFLOAT4X4 **world** = XMFLOAT4X4()

10.45.1 Detailed Description

A transform class that contains getters and setters.

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

- [CTransform.h](#)
- CTransform.cpp

10.46 CUIManager Class Reference

Static Public Member Functions

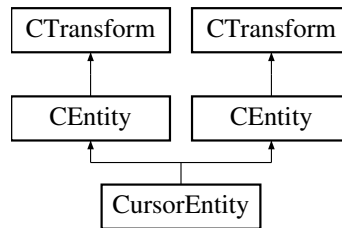
- static class [CWidget_Canvas](#) * **AddCanvas** (class [CWidget_Canvas](#) *Canvas, std::string ID)
- static void **HideAllCanvases** ()
- static class [CWidget_Canvas](#) * **GetCanvas** (std::string ID)
- static void **ClearAllCanvases** ()
- static void **UpdateUIOrigin** ([Vector3](#) Pos)

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

- CUIManager.h
- CUIManager.cpp

10.47 CursorEntity Class Reference

Inheritance diagram for CursorEntity:



Public Member Functions

- virtual void [Update](#) (float deltaTime) override
Updated automatically every single frame.
- virtual void [Update](#) (float deltaTime) override
Updated automatically every single frame.

Additional Inherited Members

10.47.1 Member Function Documentation

10.47.1.1 Update() [1/2]

```
void CursorEntity::Update (
    float deltaTime ) [override], [virtual]
```

Updated automatically every single frame.

Implements [CEntity](#).

10.47.1.2 Update() [2/2]

```
virtual void CursorEntity::Update (
    float deltaTime ) [override], [virtual]
```

Updated automatically every single frame.

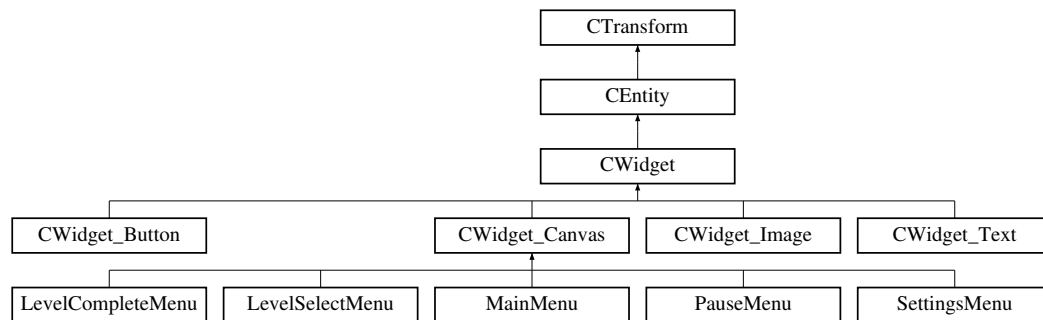
Implements [CEntity](#).

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

- CerberusTools/CursorEntity.h
- Necrodoggiecon/Game/CursorEntity.h
- CerberusTools/CursorEntity.cpp
- Necrodoggiecon/Game/CursorEntity.cpp

10.48 CWidget Class Reference

Inheritance diagram for CWidget:



Public Member Functions

- **CWidget * GetParent ()**
- **const std::vector< CWidget * > GetChildren ()**
- **virtual void SetWidgetTransform (Vector2 Position, Vector2 Anchor, int ZOrder)**
Sets the widgets transform, this is overridden by child classes.
- **virtual void SetVisibility (bool IsVisible)**
Sets the visibility of the current widget and all child components.
- **void AddChild (CWidget *NewChild)**
Adds a widget to this object.
- **void RemoveAllChildren ()**
Removes all children from this object and destroys them.
- **void UpdateWidgetOrigin (Vector3 Pos)**

Protected Attributes

- **bool WidgetIsVisible = true**

Additional Inherited Members

10.48.1 Member Function Documentation

10.48.1.1 AddChild()

```
void CWidget::AddChild (
    CWidget * NewChild )
```

Adds a widget to this object.

Parameters

<i>NewChild</i>	The new child object.
-----------------	-----------------------

10.48.1.2 SetVisibility()

```
void CWidget::SetVisibility (
    bool IsVisible ) [virtual]
```

Sets the visibility of the current widget and all child components.

This function is overridden in child classes.

Parameters

<i>IsVisible</i>	Should the widget render
------------------	--------------------------

Reimplemented in [CWidget_Button](#), [CWidget_Canvas](#), [CWidget_Image](#), and [CWidget_Text](#).

10.48.1.3 SetWidgetTransform()

```
void CWidget::SetWidgetTransform (
    Vector2 Position,
    Vector2 Anchor,
    int ZOrder ) [virtual]
```

Sets the widgets transform, this is overridden by child classes.

Parameters

<i>Position</i>	Sets position on screen
<i>Anchor</i>	Sets screen anchor
<i>ZOrder</i>	Sets the Z-Order

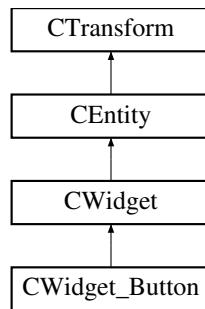
Reimplemented in [CWidget_Button](#), [CWidget_Image](#), and [CWidget_Text](#).

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

- [CWidget.h](#)
- [CWidget.cpp](#)

10.49 CWidget_Button Class Reference

Inheritance diagram for [CWidget_Button](#):



Public Member Functions

- void [SetText](#) (std::string TextBody)
Sets the button text.
- void [SetButtonSize](#) ([Vector2](#) Size)
Sets the button size, does not currently affect text.
- void [SetTexture](#) (std::string filePath)
Sets the button texture.
- virtual void [SetWidgetTransform](#) ([Vector2](#) Position, [Vector2](#) Anchor, int ZOrder)
Sets the widget transform on screen.
- virtual void [Update](#) (float deltaTime) override
Updated automatically every single frame.
- virtual void **OnButtonPressed** ()
On button Pressed event.
- virtual void **OnButtonReleased** ()
- virtual void **OnButtonHoverStart** ()
- virtual void **OnButtonHoverEnd** ()
- virtual void [SetVisibility](#) (bool IsVisible)
Sets the visibility of the current widget and all child components.
- void **IsButtonFocused** ([Vector2](#) mPos)
- void **ButtonPressed** (bool buttonPressed)
- void [Bind_OnButtonPressed](#) (std::function< void()> functionToBind)
Binds a function to this button event.
- void [Bind_OnButtonReleased](#) (std::function< void()> functionToBind)
Binds a function to this button event.
- void [Bind_HoverStart](#) (std::function< void()> functionToBind)
Binds a function to this button event.
- void [Bind_HoverEnd](#) (std::function< void()> functionToBind)
Binds a function to this button event.
- class [CSpriteComponent](#) * **GetSprite** ()
- class [CTextRenderComponent](#) * **GetText** ()
- bool **ButtonHasFocus** ()

Additional Inherited Members

10.49.1 Member Function Documentation

10.49.1.1 Bind_HoverEnd()

```
void CWidget_Button::Bind_HoverEnd (
    std::function< void()> functionToBind )    [inline]
```

Binds a function to this button event.

Parameters

<i>functionToBind</i>	The function to be bound, to bind a function use std::bind(&ClassName::FunctionName, ObjectReference)
-----------------------	---

10.49.1.2 Bind_HoverStart()

```
void CWidget_Button::Bind_HoverStart (
    std::function< void()> functionToBind )    [inline]
```

Binds a function to this button event.

Parameters

<i>functionToBind</i>	The function to be bound, to bind a function use std::bind(&ClassName::FunctionName, ObjectReference)
-----------------------	---

10.49.1.3 Bind_OnButtonPressed()

```
void CWidget_Button::Bind_OnButtonPressed (
    std::function< void()> functionToBind )    [inline]
```

Binds a function to this button event.

Parameters

<i>functionToBind</i>	The function to be bound, to bind a function use std::bind(&ClassName::FunctionName, ObjectReference)
-----------------------	---

10.49.1.4 Bind_OnButtonReleased()

```
void CWidget_Button::Bind_OnButtonReleased (
    std::function< void()> functionToBind )    [inline]
```

Binds a function to this button event.

Parameters

<i>functionToBind</i>	The function to be bound, to bind a function use <code>std::bind(&ClassName::FunctionName, ObjectReference)</code>
-----------------------	--

10.49.1.5 SetButtonSize()

```
void CWidget_Button::SetButtonSize (
    Vector2 Size )
```

Sets the button size, does not currently affect text.

Parameters

<i>Size</i>	
-------------	--

10.49.1.6 SetText()

```
void CWidget_Button::SetText (
    std::string TextBody )
```

Sets the button text.

Parameters

<i>TextBody</i>	
-----------------	--

10.49.1.7 SetTexture()

```
void CWidget_Button::SetTexture (
    std::string filePath )
```

Sets the button texture.

Parameters

<i>filePath</i>	
-----------------	--

10.49.1.8 SetVisibility()

```
void CWidget_Button::SetVisibility (
    bool IsVisible ) [virtual]
```

Sets the visibility of the current widget and all child components.

This function is overridden in child classes.

Parameters

<i>IsVisible</i>	Should the widget render
------------------	--------------------------

Reimplemented from [CWidget](#).

10.49.1.9 SetWidgetTransform()

```
void CWidget_Button::SetWidgetTransform (
    Vector2 Position,
    Vector2 Anchor,
    int ZOrder ) [virtual]
```

Sets the widget transform on screen.

Overriden from [CWidget](#).

Parameters

<i>Position</i>	Position on screen, relative to anchor
<i>Anchor</i>	Anchor position on screen
<i>ZOrder</i>	Z-Order

Reimplemented from [CWidget](#).

10.49.1.10 Update()

```
void CWidget_Button::Update (
    float deltaTime ) [override], [virtual]
```

Updated automatically every single frame.

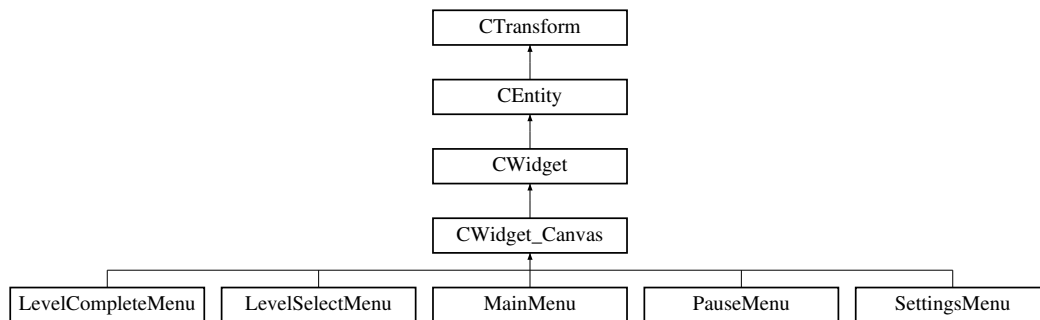
Implements [CEntity](#).

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

- [CWidget_Button.h](#)
- [CWidget_Button.cpp](#)

10.50 CWidget_Canvas Class Reference

Inheritance diagram for CWidget_Canvas:



Public Member Functions

- virtual void [InitialiseCanvas](#) ()
Initialises the canvas.
- virtual void [Update](#) (float deltaTime) override
Updated automatically every single frame.
- [Vector2](#) [GetMousePosition](#) ()
Get position of the mouse on screen.
- class [CWidget_Button](#) * [CreateButton](#) ([Vector2](#) Position, [Vector2](#) Anchor, std::string &ButtonName, int ZOrder)
Creates a Button Widget inside the canvas.
- class [CWidget_Image](#) * [CreateImage](#) ([Vector2](#) Position, [Vector2](#) Anchor, int ZOrder)
Creates an Image Widget inside the canvas.
- class [CWidget_Text](#) * [CreateText](#) ([Vector2](#) Position, [Vector2](#) Anchor, int ZOrder, std::string &Text)
Creates a Text Widget inside the canvas.
- virtual void [SetVisibility](#) (bool isVisible)
Sets the visibility of this canvas and all children.

Protected Attributes

- std::vector< class [CWidget_Button](#) * > **buttonList**
List of all buttons instantiated by this canvas, used to activate their events when required.
- bool **mouseReleased**
- bool **mousePressed**

Additional Inherited Members

10.50.1 Member Function Documentation

10.50.1.1 GetMousePosition()

```
Vector2 CWidget_Canvas::GetMousePosition ( )
```

Get position of the mouse on screen.

Returns

10.50.1.2 InitialiseCanvas()

```
void CWidget_Canvas::InitialiseCanvas ( ) [virtual]
```

Initialises the canvas.

Instantiate all widgets inside this function

10.50.1.3 SetVisibility()

```
void CWidget_Canvas::SetVisibility (
    bool IsVisible ) [virtual]
```

Sets the visibility of this canvas and all children.

Parameters

<i>IsVisible</i>	
------------------	--

Reimplemented from [CWidget](#).

10.50.1.4 Update()

```
void CWidget_Canvas::Update (
    float deltaTime ) [override], [virtual]
```

Updated automatically every single frame.

Implements [CEntity](#).

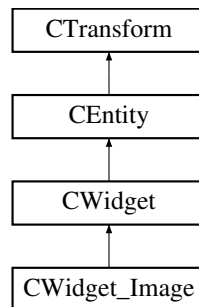
Reimplemented in [PauseMenu](#), and [SettingsMenu](#).

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

- [CWidget_Canvas.h](#)
- [CWidget_Canvas.cpp](#)

10.51 CWidget_Image Class Reference

Inheritance diagram for CWidget_Image:



Public Member Functions

- virtual void [Update](#) (float deltaTime) override
Updated automatically every single frame.
- virtual void [SetWidgetTransform](#) ([Vector2](#) Position, [Vector2](#) Anchor, int ZOrder)
Sets widget transform on screen.
- class [CSpriteComponent](#) * [GetSprite](#) ()
- class [CTextRenderComponent](#) * [GetText](#) ()
- void [SetSpriteData](#) ([Vector2](#) SpriteSize, std::string filePath)
- virtual void [SetVisibility](#) (bool isVisible)
Sets the visibility of the current widget and all child components.

Protected Attributes

- class [CSpriteComponent](#) * **sprite** = nullptr
- class [CTextRenderComponent](#) * **textRenderer** = nullptr

Additional Inherited Members

10.51.1 Member Function Documentation

10.51.1.1 SetVisibility()

```
void CWidget_Image::SetVisibility (
    bool isVisible ) [virtual]
```

Sets the visibility of the current widget and all child components.

This function is overridden in child classes.

Parameters

<i>IsVisible</i>	Should the widget render
------------------	--------------------------

Reimplemented from [CWidget](#).

10.51.1.2 SetWidgetTransform()

```
void CWidget_Image::SetWidgetTransform (
    Vector2 Position,
    Vector2 Anchor,
    int ZOrder ) [virtual]
```

Sets widget transform on screen.

Overriden from parent.

Parameters

<i>Position</i>	
<i>Anchor</i>	
<i>ZOrder</i>	

Reimplemented from [CWidget](#).

10.51.1.3 Update()

```
void CWidget_Image::Update (
    float deltaTime ) [override], [virtual]
```

Updated automatically every single frame.

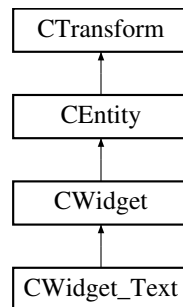
Implements [CEntity](#).

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

- [CWidget_Image.h](#)
- [CWidget_Image.cpp](#)

10.52 CWidget_Text Class Reference

Inheritance diagram for CWidget_Text:



Public Member Functions

- virtual void [Update](#) (float deltaTime) override
Updated automatically every single frame.
- virtual void [SetWidgetTransform](#) ([Vector2](#) Position, [Vector2](#) Anchor, int ZOrder)
Sets the widgets transform, this is overridden by child classes.
- virtual void [SetVisibility](#) (bool IsVisible)
Sets the visibility of the current widget and all child components.
- class [CTextRenderComponent](#) * **GetText** ()

Protected Attributes

- class [CTextRenderComponent](#) * **textRenderer** = nullptr

Additional Inherited Members

10.52.1 Member Function Documentation

10.52.1.1 SetVisibility()

```
void CWidget_Text::SetVisibility (
    bool IsVisible ) [virtual]
```

Sets the visibility of the current widget and all child components.

This function is overridden in child classes.

Parameters

<i>IsVisible</i>	Should the widget render
------------------	--------------------------

Reimplemented from [CWidget](#).

10.52.1.2 SetWidgetTransform()

```
void CWidget_Text::SetWidgetTransform (
    Vector2 Position,
    Vector2 Anchor,
    int ZOrder ) [virtual]
```

Sets the widgets transform, this is overridden by child classes.

Parameters

<i>Position</i>	Sets position on screen
<i>Anchor</i>	Sets screen anchor
<i>ZOrder</i>	Sets the Z-Order

Reimplemented from [CWidget](#).

10.52.1.3 Update()

```
void CWidget_Text::Update (
    float deltaTime ) [override], [virtual]
```

Updated automatically every single frame.

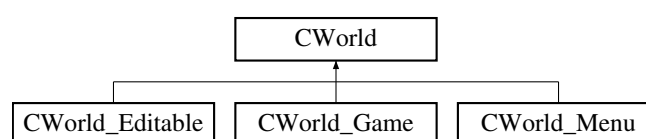
Implements [CEntity](#).

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

- CWidget_Text.h
- CWidget_Text.cpp

10.53 CWorld Class Reference

Inheritance diagram for CWorld:



Public Member Functions

- **CWorld** (int Slot)
- int **GetMapSlot** ()
- virtual void **LoadWorld** (int Slot)
- virtual void **SetupWorld** ()
- virtual void **UnloadWorld** ()
- virtual void **ReloadWorld** ()
- virtual void **DestroyWorld** ()
- [CTile](#) * **GetTileByID** (int ID)
- std::vector< [CTile](#) * > **GetAllWalkableTiles** ()
- std::vector< [CTile](#) * > **GetAllObstacleTiles** ()
- void **BuildNavigationGrid** ()
- void **AddEntityToList** (class [CEntity](#) *NewEntity)

Protected Member Functions

- virtual void **LoadEntities** (int Slot)
- [Vector3](#) **IndexToGrid** (int ID)
- int **GridToIndex** ([Vector2](#) Position)

Protected Attributes

- int [mapSize](#)
- [CTile](#) * **tileContainer** [mapScale *mapScale]
- int **mapSlot**
- std::vector< [CEntity](#) * > **EntityList**
- [Vector2](#) **StartPos**

10.53.1 Member Data Documentation

10.53.1.1 mapSize

```
int CWorld::mapSize [protected]
```

Initial value:

```
=
```

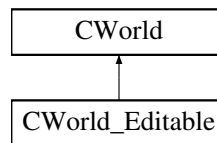
```
mapScale * mapScale
```

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

- CWorld.h
- CWorld.cpp

10.54 CWorld_Editable Class Reference

Inheritance diagram for CWorld_Editable:



Public Member Functions

- EditOperationMode **GetOperationMode** ()
- void **SetOperationMode** (EditOperationMode mode)
- void **SetEntityID** (int ID)
- void **QueueCell** ([Vector2](#) Cell)
- void **ToggleCellQueueLock** (bool setLock)
- void **ClearQueue** ()
- void **PerformOperation** ([Vector2](#) A, [Vector2](#) B)
- void **PerformOperation_ClearSpace** ()
- virtual void **LoadWorld** (int Slot) override
- virtual void **UnloadWorld** () override
- virtual void **SetupWorld** ()
- void **SaveWorld** (int Slot)
- void **EditWorld** (int Slot)
- void **NewWorld** (int Slot)
- void **ToggleDebugMode** (bool isDebug)
- void **UpdateEditorViewport** ()
- EditorEntityType **GetInspectedItemType** ()
- [CT_EditorEntity](#) * **GetInspectedItem_Standard** ()
- class [CT_EditorEntity_Enemy](#) * **GetInspectedItem_Enemy** ()
- [CT_EditorEntity_Waypoint](#) * **GetInspectedItem_Waypoint** ()
- [CT_EditorEntity_WeaponHolder](#) * **GetInspectedItem_WeaponHolder** ()
- void **ShouldInspectEntity** ([Vector2](#) MousePos)
- void **MoveSelectedEntity** ([Vector3](#) Position)
- void **RemoveSelectedEntity** ()

Protected Member Functions

- void **AdditiveBox** ([Vector2](#) A, [Vector2](#) B)
- void **SubtractiveBox** ([Vector2](#) A, [Vector2](#) B)
- void **AdditiveBox_Scale** ([Vector2](#) A, [Vector2](#) B)
- void **SubtractiveBox_Scale** ([Vector2](#) A, [Vector2](#) B)
- void **ClearSpace** ()
- void **Additive_Cell** ([Vector2](#) A)
- void **Subtractive_Cell** ([Vector2](#) A)
- void **AddEditorEntity_EnemyCharacter** ([Vector2](#) Position, int Slot)
- void **AddEditorEntity_Decoration** ([Vector2](#) Position, int Slot)
- void **AddEditorEntity_Waypoint** ([Vector2](#) Position)
- void **AddEditorEntity_Prop** (int Slot)
- void **AddEditorEntity_WeaponHolder** ([Vector2](#) Position)
- void **GeneratePropList** ()

Additional Inherited Members

10.54.1 Member Function Documentation

10.54.1.1 LoadWorld()

```
void CWorld_Editable::LoadWorld (
    int Slot ) [override], [virtual]
```

Reimplemented from [CWorld](#).

10.54.1.2 SetupWorld()

```
void CWorld_Editable::SetupWorld ( ) [virtual]
```

Reimplemented from [CWorld](#).

10.54.1.3 UnloadWorld()

```
void CWorld_Editable::UnloadWorld ( ) [override], [virtual]
```

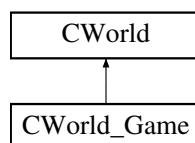
Reimplemented from [CWorld](#).

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

- CWorld_Edit.h
- CWorld_Edit.cpp

10.55 CWorld_Game Class Reference

Inheritance diagram for CWorld_Game:



Public Member Functions

- [CWorld_Game](#) (int Slot)
Constructor, automatically loads world based on provided slot.
- virtual void [SetupWorld](#) ()
- virtual void [UnloadWorld](#) ()
- virtual void [ReloadWorld](#) ()
- virtual void **LoadEnemyUnits** (int Slot)
- virtual void [LoadEntities](#) (int Slot) override

Additional Inherited Members

10.55.1 Constructor & Destructor Documentation

10.55.1.1 CWorld_Game()

```
CWorld_Game::CWorld_Game (
    int Slot )
```

Constructor, automatically loads world based on provided slot.

Parameters

<i>Slot</i>	Determines which level to load.
-------------	---------------------------------

10.55.2 Member Function Documentation

10.55.2.1 LoadEntities()

```
void CWorld_Game::LoadEntities (
    int Slot ) [override], [virtual]
```

Reimplemented from [CWorld](#).

10.55.2.2 ReloadWorld()

```
void CWorld_Game::ReloadWorld ( ) [virtual]
```

Reimplemented from [CWorld](#).

10.55.2.3 SetupWorld()

```
void CWorld_Game::SetupWorld ( ) [virtual]
```

Reimplemented from [CWorld](#).

10.55.2.4 UnloadWorld()

```
void CWorld_Game::UnloadWorld ( ) [virtual]
```

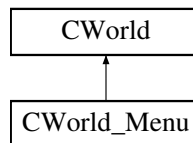
Reimplemented from [CWorld](#).

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

- CWorld_Game.h
- CWorld_Game.cpp

10.56 CWorld_Menu Class Reference

Inheritance diagram for CWorld_Menu:



Additional Inherited Members

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

- CWorld_Menu.h
- CWorld_Menu.cpp

10.57 CWorldManager Class Reference

Static Public Member Functions

- static void [LoadWorld](#) (int Slot, bool bEditorMode)
Loads in a level by slot, automatically unloads the previous level.
- static void [LoadWorld](#) ([CWorld](#) *World)
Loads an override object of world, this is primarily used by the game to instantiate child class variants of the existing level class.
- static void [LoadWorld](#) ([CWorld_Editable](#) *World)
Edit world variant of the load world override.
- static void **ReloadWorld** ()
- static class [CWorld](#) * **GetWorld** ()
- static class [CWorld_Editable](#) * **GetEditorWorld** ()

10.57.1 Member Function Documentation

10.57.1.1 LoadWorld() [1/3]

```
void CWorldManager::LoadWorld (
    CWorld * World ) [static]
```

Loads an override object of world, this is primarily used by the game to instantiate child class variants of the existing level class.

Parameters

<i>World</i>	
--------------	--

10.57.1.2 LoadWorld() [2/3]

```
void CWorldManager::LoadWorld (
    CWorld_Editable * World ) [static]
```

Edit world variant of the load world override.

Parameters

<i>World</i>	
--------------	--

10.57.1.3 LoadWorld() [3/3]

```
void CWorldManager::LoadWorld (
    int Slot,
    bool bEditorMode ) [static]
```

Loads in a level by slot, automatically unloads the previous level.

Can determine whether the level loaded is an editor version or standard.

Parameters

<i>Slot</i>	
<i>bEditorMode</i>	

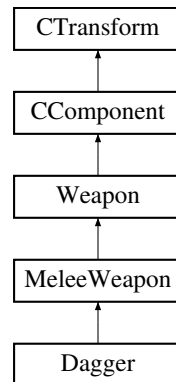
The documentation for this class was generated from the following files:

- CWorldManager.h

- CWorldManager.cpp

10.58 Dagger Class Reference

Inheritance diagram for Dagger:



Additional Inherited Members

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

- [Dagger.h](#)
- [Dagger.cpp](#)

10.59 Debug Class Reference

Static Public Member Functions

- static void [SetVisibility](#) (bool value)
Sets the visibility of the debug output console.
- static bool [GetVisibility](#) ()
Returns the visibility of the debug output console.
- static void [SetLogging](#) (bool value)
Sets the ability to log to the debug output console.
- static bool [GetLogging](#) ()
Returns whether you can log to the debug output console.
- template<typename ... Args>
static void [Log](#) (const char *fmt, Args ... args) IM_FMTARGS(2)
Logs a formatted string to the output console.
- template<typename ... Args>
static void [LogError](#) (const char *fmt, Args ... args) IM_FMTARGS(2)
Logs a formatted string to the output console in red to indicate a error.
- template<typename ... Args>
static void [LogHResult](#) (HRESULT hr, const char *fmt, Args ... args) IM_FMTARGS(2)
Logs a formatted string to the output console with support for HRESULT checking.
- static [DebugOutput](#) * [getOutput](#) ()
Returns the output console if it exists.

10.59.1 Member Function Documentation

10.59.1.1 GetLogging()

```
static bool Debug::GetLogging ( ) [inline], [static]
```

Returns whether you can log to the debug output console.

Returns

whether logging is disabled / enabled.

10.59.1.2 getOutput()

```
static DebugOutput * Debug::getOutput ( ) [inline], [static]
```

Returns the output console if it exists.

Returns

a pointer to the output console.

10.59.1.3 GetVisibility()

```
static bool Debug::GetVisibility ( ) [inline], [static]
```

Returns the visibility of the debug output console.

Returns

the visiblity of the debug output console.

10.59.1.4 Log()

```
template<typename ... Args>  
static void Debug::Log (  
    const char * fmt,  
    Args ... args ) [inline], [static]
```

Logs a formatted string to the output console.

Parameters

<i>fmt</i>	the string you wish to print with formatting.
<i>args</i>	the extra formatted arguments you wish to put inside the string.

10.59.1.5 LogError()

```
template<typename ... Args>
static void Debug::LogError (
    const char * fmt,
    Args ... args ) [inline], [static]
```

Logs a formatted string to the output console in red to indicate a error.

Parameters

<i>fmt</i>	the string you wish to print with formatting.
<i>args</i>	the extra formatted arguments you wish to put inside the string.

10.59.1.6 LogHRESULT()

```
template<typename ... Args>
static void Debug::LogHRESULT (
    HRESULT hr,
    const char * fmt,
    Args ... args ) [inline], [static]
```

Logs a formatted string to the output console with support for HRESULT checking.

Parameters

<i>hr</i>	the HRESULT you wish to check before outputting error or success.
<i>fmt</i>	the string you wish to print with formatting.
<i>args</i>	the extra formatted arguments you wish to put inside the string.

10.59.1.7 SetLogging()

```
static void Debug::SetLogging (
    bool value ) [inline], [static]
```

Sets the ability to log to the debug output console.

Parameters

<i>value</i>	allow/disallow logging to the debug console.
--------------	--

10.59.1.8 SetVisibility()

```
static void Debug::SetVisibility (
    bool value ) [inline], [static]
```

Sets the visibility of the debug output console.

Parameters

<i>value</i>	show/hide the debug console.
--------------	------------------------------

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

- [Debug.h](#)
- Debug.cpp

10.60 DebugOutput Class Reference

Public Member Functions

- `ImVector< char * > getItems ()`
- `void ClearLog ()`
- `void AddLog (const char *fmt,...) IM_FMTARGS(2)`
- `void render ()`

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

- DebugOutput.h

10.61 Dialogue Struct Reference

Public Member Functions

- `Dialogue (std::string name, std::string dialogue)`

Public Attributes

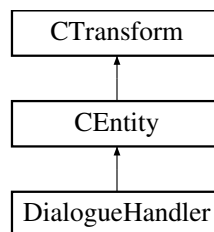
- `std::string` **name**
- `std::string` **dialogue**

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

- Dialogue.h

10.62 DialogueHandler Class Reference

Inheritance diagram for DialogueHandler:



Static Public Member Functions

- static void [SetDialogue](#) (const `std::string` &name, const `std::string` &dialogue)
Function to set the dialogue that should display.
- static void [LoadDialogue](#) (const `std::string` &jsonPath, const `std::string` &dialogueName)
Function to load dialogue from a json file.
- static void [AdvanceDialogue](#) ()
Function used to move dialogue to the next stage.
- static void **CloseDialogue** ()
Function to clear the text on the dialogue UI and disable drawing.
- static void **SetInstantDisplay** (bool _instantDisplay)

Additional Inherited Members

10.62.1 Member Function Documentation

10.62.1.1 AdvanceDialogue()

```
void DialogueHandler::AdvanceDialogue ( ) [static]
```

Function used to move dialogue to the next stage.

Will either complete the current page, go to the next page, load the next piece of dialogue or close the dialogue UI

10.62.1.2 LoadDialogue()

```
void DialogueHandler::LoadDialogue (
    const std::string & jsonPath,
    const std::string & dialogueName ) [static]
```

Function to load dialogue from a json file.

Will the call the SetDialogue function using the first instance of dialogue in the json file. Called Like [DialogueHandler::LoadDialogue](#)("Resources/Game/Dialogue.json", "TestDialogue")

10.62.1.3 SetDialogue()

```
void DialogueHandler::SetDialogue (
    const std::string & name,
    const std::string & dialogue ) [static]
```

Function to set the dialogue that should display.

Calls the SetName and SetText functions on the dialogueUI

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

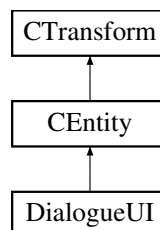
- DialogueHandler.h
- [DialogueHandler.cpp](#)

10.63 DialogueUI Class Reference

Class that handles displaying text in the dialogue window.

```
#include <DialogueUI.h>
```

Inheritance diagram for DialogueUI:



Public Member Functions

- **DialogueUI** ()
Constructor - Initialises all of the UI elements including text components and backgrounds.
- virtual void **Update** (float deltaTime) override
Inherited Function - Used to add characters to the display over time.
- void **SetText** (const std::string &newText, bool instantDisplay)
Function used to set the text that will display in the dialogue box.
- void **SetName** (const std::string &newName)
Function used to set the name text above the dialogue box.
- void **ClearText** ()
Function used to clear the text being displayed in the dialogue box.
- void **Complete** ()
- void **CompletePage** ()
Function used to instantly display as much dialogue from the current section of dialogue on the screen as possible.
- bool **IsUpdating** ()
- bool **IsComplete** ()
Function used to check whether the current section of dialogue is complete.
- void **Advance** ()
Function used to advance the current section of dialogue.
- void **ToggleDrawing** (bool shouldDraw)
Function used to enable and disable drawing of the dialogue box.
- int **GetReserveCharacterCount** ()

Additional Inherited Members

10.63.1 Detailed Description

Class that handles displaying text in the dialogue window.

10.63.2 Member Function Documentation

10.63.2.1 Advance()

```
void DialogueUI::Advance ( )
```

Function used to advance the current section of dialogue.

Should only be called once the dialogue box is full.

10.63.2.2 SetName()

```
void DialogueUI::SetName (  
    const std::string & newName )
```

Function used to set the name text above the dialogue box.

Parameters

<i>newName</i>	- The new name that should be displayed.
----------------	--

10.63.2.3 SetText()

```
void DialogueUI::SetText (
    const std::string & newText,
    bool instantDisplay )
```

Function used to set the text that will display in the dialogue box.

Parameters

<i>newText</i>	- The new text (section of dialogue) that will display.
<i>instantDisplay</i>	- Whether the text should update instantly or overtime

10.63.2.4 ToggleDrawing()

```
void DialogueUI::ToggleDrawing (
    bool shouldDraw )
```

Function used to enable and disable drawing of the dialogue box.

Parameters

<i>shouldDraw</i>	- Whether the dialogue UI should draw or not.
-------------------	---

10.63.2.5 Update()

```
void DialogueUI::Update (
    float deltaTime ) [override], [virtual]
```

Inherited Function - Used to add characters to the display over time.

Implements [CEntity](#).

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

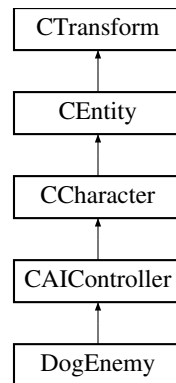
- DialogueUI.h
- [DialogueUI.cpp](#)

10.64 DogEnemy Class Reference

Class for the dog enemy.

```
#include <DogEnemy.h>
```

Inheritance diagram for DogEnemy:



Public Member Functions

- virtual void [Update](#) (float deltaTime) override
- virtual void [ChasePlayer](#) ([CCharacter](#) *player) override
Seek towards the player and switch to attacking once in range.
- virtual void [AttackEnter](#) ([CCharacter](#) *player) override
Get the target position to dash towards.
- virtual void [AttackPlayer](#) ([CCharacter](#) *player, float deltaTime) override
If not on cooldown then charge up a dash attack and then dash at the target position.

Protected Member Functions

- virtual void [OnDeath](#) () override
- virtual void [OnHit](#) (const std::string &hitSound) override

Additional Inherited Members

10.64.1 Detailed Description

Class for the dog enemy.

The dog will dash at the player once it's within attack range.

10.64.2 Member Function Documentation

10.64.2.1 AttackEnter()

```
void DogEnemy::AttackEnter (
    CCharacter * player ) [override], [virtual]
```

Get the target position to dash towards.

Parameters

<i>player</i>	Player to target for an attack.
---------------	---------------------------------

Reimplemented from [CAIController](#).

10.64.2.2 AttackPlayer()

```
void DogEnemy::AttackPlayer (
    CCharacter * player,
    float deltaTime ) [override], [virtual]
```

If not on cooldown then charge up a dash attack and then dash at the target position.

Parameters

<i>player</i>	Player to attack.
---------------	-------------------

Reimplemented from [CAIController](#).

10.64.2.3 ChasePlayer()

```
void DogEnemy::ChasePlayer (
    CCharacter * player ) [override], [virtual]
```

Seek towards the player and switch to attacking once in range.

Parameters

<i>player</i>	Player to seek towards.
---------------	-------------------------

Reimplemented from [CAIController](#).

10.64.2.4 OnDeath()

```
void DogEnemy::OnDeath ( ) [override], [protected], [virtual]
```

Reimplemented from [CAIController](#).

10.64.2.5 OnHit()

```
void DogEnemy::OnHit (
    const std::string & hitSound ) [override], [protected], [virtual]
```

Reimplemented from [CAIController](#).

10.64.2.6 Update()

```
void DogEnemy::Update (
    float deltaTime ) [override], [virtual]
```

Parameters

<i>deltaTime</i>	
------------------	--

Reimplemented from [CAIController](#).

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

- [DogEnemy.h](#)
- [DogEnemy.cpp](#)

10.65 Engine Struct Reference

Static Public Member Functions

- static bool **Start** (HINSTANCE hInstance, int nCmdShow, WNDPROC wndProc)
- static void **RenderUpdateLoop** ()
- static LRESULT **ReadMessage** (HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
- static void **Stop** ()
- static void **SetRenderCamera** ([CCameraComponent](#) *cam)
- template<class T >
static std::vector< T * > **GetEntityOfType** ()
- static void **DestroyEntity** ([CEntity](#) *targetEntity)
- template<class T >
static T * **CreateEntity** ()

Static Public Attributes

- static HINSTANCE **instanceHandle**
- static HWND **windowHandle**
- static unsigned int **windowWidth** = 1280
- static unsigned int **windowHeight** = 720
- static D3D_DRIVER_TYPE **driverType** = D3D_DRIVER_TYPE_NULL
- static D3D_FEATURE_LEVEL **featureLevel** = D3D_FEATURE_LEVEL_11_0
- static ID3D11Device * **device**

- static ID3D11DeviceContext * **deviceContext**
- static XMMATRIX **projMatrixUI** = XMMatrixIdentity()
- static bool **paused** = false

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

- Engine.h
- Engine.cpp

10.66 EntityManager Class Reference

Static class for tracking entities and components while accommodating translucency.

```
#include <EntityManager.h>
```

Static Public Member Functions

- static void **AddEntity** (class [CEntity](#) *entityToAdd)
Adds the input entity to the internal vector.
- static void **RemoveEntity** (const class [CEntity](#) *entityToRemove)
Removes the input entity to the internal vector.
- static void **AddComponent** (class [CComponent](#) *compToAdd)
Adds the input component to the internal containers based on translucency boolean in [CComponent](#).
- static void **RemoveComponent** (const class [CComponent](#) *compToRemove)
Removes the input component to the internal containers based on translucency boolean in [CComponent](#).
- static void **SortTranslucentComponents** ()
Sorts the translucent components container ready for drawing.
- static const std::vector< class [CEntity](#) * > * **GetEntitiesVector** ()
- static const std::vector< class [CComponent](#) * > * **GetOpaqueCompsVector** ()
- static const std::vector< class [CComponent](#) * > * **GetTranslucentCompsVector** ()

10.66.1 Detailed Description

Static class for tracking entities and components while accommodating translucency.

10.66.2 Member Function Documentation

10.66.2.1 RemoveComponent()

```
void EntityManager::RemoveComponent (
    const class CComponent * compToRemove ) [static]
```

Removes the input component to the internal containers based on translucency boolean in [CComponent](#).

Note: does NOT delete the component.

10.66.2.2 RemoveEntity()

```
void EntityManager::RemoveEntity (
    const class CEntity * entityToRemove ) [static]
```

Removes the input entity to the internal vector.

Note: does NOT delete the entity.

10.66.2.3 SortTranslucentComponents()

```
void EntityManager::SortTranslucentComponents ( ) [static]
```

Sorts the translucent components container ready for drawing.

This is done automatically in the engine's draw function so DON'T call this.

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

- [EntityManager.h](#)
- [EntityManager.cpp](#)

10.67 EventSystem Class Reference

Static Public Member Functions

- static void [AddListener](#) (std::string eventID, std::function< void()> functionToAdd)
Adds a listener to a specific event ID.
- static void [TriggerEvent](#) (std::string eventID)
Triggers the event of specified ID.

10.67.1 Member Function Documentation

10.67.1.1 AddListener()

```
void EventSystem::AddListener (
    std::string eventID,
    std::function< void()> functionToAdd ) [static]
```

Adds a listener to a specific event ID.

Parameters

<i>eventID</i>	eventID that will trigger this event
<i>functionToAdd</i>	function that will be triggered when the event is called.

10.67.1.2 TriggerEvent()

```
void EventSystem::TriggerEvent (
    std::string eventID ) [static]
```

Triggers the event of specified ID.

Parameters

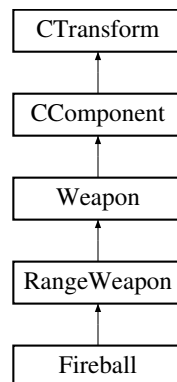
<i>eventID</i>	eventID of the specific event that is triggered.
----------------	--

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

- [EventSystem.h](#)
- [EventSystem.cpp](#)

10.68 Fireball Class Reference

Inheritance diagram for Fireball:



Additional Inherited Members

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

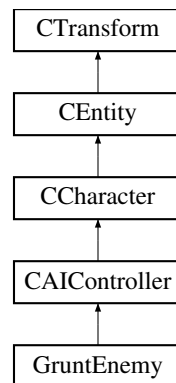
- [Fireball.h](#)
- [Fireball.cpp](#)

10.69 GruntEnemy Class Reference

Class for the Grunt enemy.

```
#include <GruntEnemy.h>
```

Inheritance diagram for GruntEnemy:



Public Member Functions

- virtual void [ChasePlayer](#) ([CCharacter](#) *player) override
Seek towards the player and if in range go to the attack state.
- virtual void [AttackPlayer](#) ([CCharacter](#) *player, float deltaTime) override
Fire the weapon that it is holding.

Protected Member Functions

- virtual void [OnDeath](#) () override
- virtual void [OnHit](#) (const std::string &hitSound) override
- virtual void [Update](#) (float deltaTime) override
- void [UpdateWeaponSprite](#) ()

Additional Inherited Members

10.69.1 Detailed Description

Class for the Grunt enemy.

This enemy will use the weapon it is holding when it gets in range of the player.

10.69.2 Member Function Documentation

10.69.2.1 AttackPlayer()

```
void GruntEnemy::AttackPlayer (
    CCharacter * player,
    float deltaTime ) [override], [virtual]
```

Fire the weapon that it is holding.

Parameters

<i>player</i>	Player to attack.
---------------	-------------------

Reimplemented from [CAIController](#).

10.69.2.2 ChasePlayer()

```
void GruntEnemy::ChasePlayer (
    CCharacter * player ) [override], [virtual]
```

Seek towards the player and if in range go to the attack state.

Parameters

<i>player</i>	
---------------	--

Reimplemented from [CAIController](#).

10.69.2.3 OnDeath()

```
void GruntEnemy::OnDeath ( ) [override], [protected], [virtual]
```

Reimplemented from [CAIController](#).

10.69.2.4 OnHit()

```
void GruntEnemy::OnHit (
    const std::string & hitSound ) [override], [protected], [virtual]
```

Reimplemented from [CAIController](#).

10.69.2.5 Update()

```
void GruntEnemy::Update (
    float deltaTime ) [override], [protected], [virtual]
```

Parameters

<i>deltaTime</i>	
------------------	--

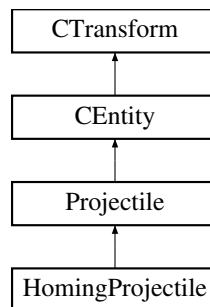
Reimplemented from [CAIController](#).

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

- [GruntEnemy.h](#)
- [GruntEnemy.cpp](#)

10.70 HomingProjectile Class Reference

Inheritance diagram for HomingProjectile:



Public Member Functions

- virtual void [Update](#) (float deltaTime)
Will make a projectile that will home into a enemy.

Additional Inherited Members

10.70.1 Member Function Documentation

10.70.1.1 Update()

```
void HomingProjectile::Update (  
    float deltaTime ) [virtual]
```

Will make a projectile that will home into a enemy.

\Homes and then Damages the target if it hit

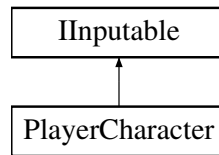
Reimplemented from [Projectile](#).

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

- [HomingProjectile.h](#)
- [HomingProjectile.cpp](#)

10.71 IInputable Class Reference

Inheritance diagram for IInputable:



Public Member Functions

- virtual void [PressedHorizontal](#) (int dir, float deltaTime)=0
- virtual void [PressedVertical](#) (int dir, float deltaTime)=0
- virtual void [PressedInteract](#) ()=0
- virtual void [PressedDrop](#) ()=0
- virtual void **Attack** ()=0
- virtual void **PressedUse** ()=0

10.71.1 Member Function Documentation

10.71.1.1 PressedDrop()

```
virtual void IInputable::PressedDrop ( ) [pure virtual]
```

Implemented in [PlayerCharacter](#).

10.71.1.2 PressedHorizontal()

```
virtual void IInputable::PressedHorizontal (
    int dir,
    float deltaTime ) [pure virtual]
```

Implemented in [PlayerCharacter](#).

10.71.1.3 PressedInteract()

```
virtual void IInputable::PressedInteract ( ) [pure virtual]
```

Implemented in [PlayerCharacter](#).

10.71.1.4 PressedVertical()

```
virtual void IInputable::PressedVertical (
    int dir,
    float deltaTime ) [pure virtual]
```

Implemented in [PlayerCharacter](#).

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

- [IInputable.h](#)

10.72 InputManager Class Reference

Public Types

- enum **Keys** {
A , **B** , **C** , **D** ,
E , **F** , **G** , **H** ,
I , **J** , **K** , **L** ,
M , **N** , **O** , **P** ,
Q , **R** , **S** , **T** ,
U , **V** , **W** , **X** ,
Y , **Z** , **Num0** , **Num1** ,
Num2 , **Num3** , **Num4** , **Num5** ,
Num6 , **Num7** , **Num8** , **Num9** ,
Escape , **LControl** , **LShift** , **LAlt** ,
LWindows , **RControl** , **RShift** , **RAlt** ,
RWindows , **Menu** , **LBracket** , **RBracket** ,
Semicolon , **Comma** , **Period** , **Slash** ,
Backslash , **Tilde** , **Equals** , **Minus** ,
Space , **Enter** , **Backspace** , **Tab** ,
PageUp , **PageDown** , **End** , **Home** ,
Insert , **Delete** , **Add** , **Subtract** ,
Multiply , **Divide** , **Left** , **Right** ,
Up , **Down** , **Numpad0** , **Numpad1** ,
Numpad2 , **Numpad3** , **Numpad4** , **Numpad5** ,
Numpad6 , **Numpad7** , **Numpad8** , **Numpad9** ,
F1 , **F2** , **F3** , **F4** ,
F5 , **F6** , **F7** , **F8** ,
F9 , **F10** , **F11** , **F12** ,
COUNT }
- enum **Mouse** { **LButton** , **RButton** , **MButton** , **MCOUNT** }

Static Public Member Functions

- static bool **IsKeyPressed** (Keys key)
\ See if the async key called was pressed
- static bool **IsKeyPressedDown** (Keys key)
\ See if the async key called was pressed down
- static bool **IsKeyReleased** (Keys key)
\ See if the async key called was released
- static bool **IsMouseButtonPressed** (Mouse mouse)
\ See if the mouse async key called was pressed
- static bool **IsMouseButtonPressedDown** (Mouse mouse)
\ See if the mouse async key called was pressed down
- static bool **IsMouseButtonReleased** (Mouse mouse)
\ See if the mouse async key called was released

Static Public Attributes

- static [Vector3](#) **mousePos** = { 0,0,0 }

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

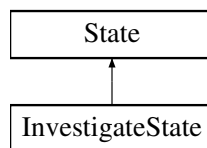
- [InputManager.h](#)
- [InputManager.cpp](#)

10.73 InvestigateState Class Reference

[State](#) for when the AI is investigating.

```
#include <State.h>
```

Inheritance diagram for InvestigateState:



Public Member Functions

- void [Enter](#) ([CAIController](#) *controller) override
- void [Update](#) ([CAIController](#) *controller, float deltaTime) override
- void [Exit](#) ([CAIController](#) *controller) override

Static Public Member Functions

- static [State](#) & **getInstance** ()

10.73.1 Detailed Description

[State](#) for when the AI is investigating.

The AI will path to the ivestigation position then enter the search state.

10.73.2 Member Function Documentation

10.73.2.1 Enter()

```
void InvestigateState::Enter (
    CAIController * controller ) [override], [virtual]
```

Reimplemented from [State](#).

10.73.2.2 Exit()

```
void InvestigateState::Exit (
    CAIController * controller ) [override], [virtual]
```

Reimplemented from [State](#).

10.73.2.3 Update()

```
void InvestigateState::Update (
    CAIController * controller,
    float deltaTime ) [override], [virtual]
```

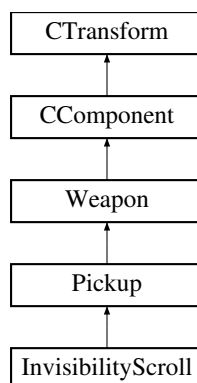
Reimplemented from [State](#).

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

- [State.h](#)
- [State.cpp](#)

10.74 InvisibilityScroll Class Reference

Inheritance diagram for InvisibilityScroll:



Additional Inherited Members

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

- [InvisibilityScroll.h](#)
- [InvisibilityScroll.cpp](#)

10.75 IO Class Reference

Static Public Member Functions

- static std::string [FindExtension](#) (const std::string &path)
Returns the extension of a file as a string.

10.75.1 Member Function Documentation

10.75.1.1 FindExtension()

```
static std::string IO::FindExtension (  
    const std::string & path )    [inline], [static]
```

Returns the extension of a file as a string.

Parameters

<i>path</i>	to a file.
-------------	------------

Returns

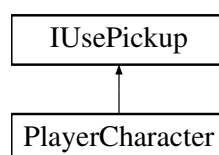
extension of file in path specified.

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

- [IO.h](#)

10.76 IUsePickup Class Reference

Inheritance diagram for IUsePickup:



Public Member Functions

- virtual void [UsePickup](#) (const std::string &pickupToUse, float activeTime)=0

10.76.1 Member Function Documentation

10.76.1.1 UsePickup()

```
virtual void IUsePickup::UsePickup (
    const std::string & pickupToUse,
    float activeTime ) [pure virtual]
```

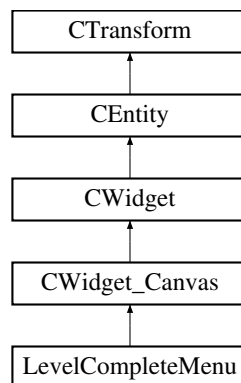
Implemented in [PlayerCharacter](#).

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

- IUsePickup.h

10.77 LevelCompleteMenu Class Reference

Inheritance diagram for LevelCompleteMenu:



Public Member Functions

- void **QuitToMenu** ()
quits back to main menu.
- void **QuitToDesktop** ()
quits game entirely.
- void **NextLevel** ()
loads next level.

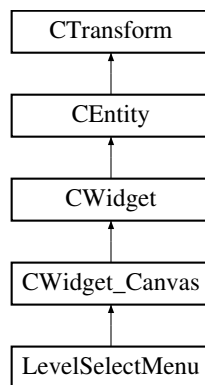
Additional Inherited Members

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

- [LevelCompleteMenu.h](#)
- [LevelCompleteMenu.cpp](#)

10.78 LevelSelectMenu Class Reference

Inheritance diagram for LevelSelectMenu:



Public Member Functions

- void **CloseMenu** ()
closes menu and reveals main menu.
- void **OpenLevelTutorial** ()
moves selected level to center.
- void **OpenLevel1** ()
moves selected level to center.
- void **OpenLevel2** ()
moves selected level to center.
- void **OpenLevel3** ()
moves selected level to center.
- void **OpenLevel4** ()
moves selected level to center.
- void **OpenLevel5** ()
moves selected level to center.
- void **OpenLevel6** ()
moves selected level to center.
- void **OpenLevel7** ()
moves selected level to center.
- void **UpdateButtonPositions** ()
offsets all level buttons to show which is selected.
- void **PlayLevel** ()
Loads the currently selected level.

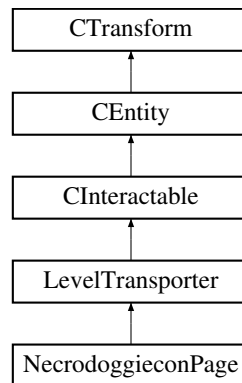
Additional Inherited Members

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

- [LevelSelectMenu.h](#)
- [LevelSelectMenu.cpp](#)

10.79 LevelTransporter Class Reference

Inheritance diagram for LevelTransporter:



Public Member Functions

- void **SetSlot** (int SlotID)
- virtual void [OnInteract](#) ()
Called when a player has interacted with the interactable.
- int **GetSlot** ()

Additional Inherited Members

10.79.1 Member Function Documentation

10.79.1.1 OnInteract()

```
void LevelTransporter::OnInteract ( ) [virtual]
```

Called when a player has interacted with the interactable.

Reimplemented from [CInteractable](#).

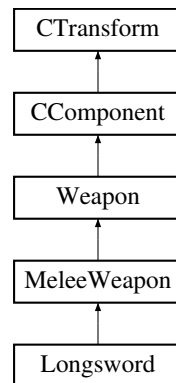
Reimplemented in [NecrodoggieconPage](#).

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

- LevelTransporter.h
- LevelTransporter.cpp

10.80 Longsword Class Reference

Inheritance diagram for Longsword:



Public Member Functions

- virtual bool [OnFire](#) ([Vector3](#) actorPos, [Vector3](#) attackDir)
Virtual override OnFire containing unique sweeping logic.

Additional Inherited Members

10.80.1 Member Function Documentation

10.80.1.1 OnFire()

```
bool Longsword::OnFire (  
    Vector3 actorPos,  
    Vector3 attackDir ) [virtual]
```

Virtual override OnFire containing unique sweeping logic.

Parameters

<i>actorPos</i>	
<i>attackDir</i>	

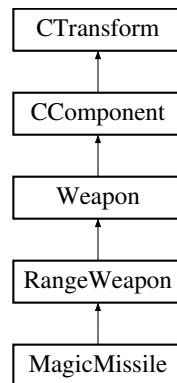
Reimplemented from [MeleeWeapon](#).

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

- [Longsword.h](#)
- Longsword.cpp

10.81 MagicMissile Class Reference

Inheritance diagram for MagicMissile:



Public Member Functions

- virtual bool [OnFire](#) ([Vector3](#) actorPos, [Vector3](#) attackDir)
Will spawn a homing projectile insaid of a normal projectile.

Additional Inherited Members

10.81.1 Member Function Documentation

10.81.1.1 OnFire()

```
bool MagicMissile::OnFire (  
    Vector3 actorPos,  
    Vector3 attackDir ) [virtual]
```

Will spawn a homing projectile insaid of a normal projectile.

\Uses the onfire to make a homing projectile insaid of the other projectile

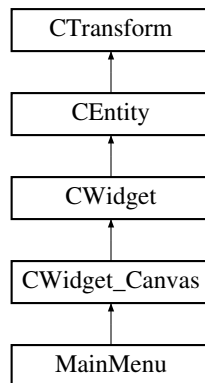
Reimplemented from [RangeWeapon](#).

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

- [MagicMissile.h](#)
- [MagicMissile.cpp](#)

10.82 MainMenu Class Reference

Inheritance diagram for MainMenu:



Public Member Functions

- void **QuitToDesktop** ()
closes game.
- void **OpenLevelSelect** ()
opens level select menu.
- void **OpenSettingsMenu** ()
opens settings menu.

Additional Inherited Members

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

- [MainMenu.h](#)
- [MainMenu.cpp](#)

10.83 MaterialPropertiesConstantBuffer Struct Reference

Public Attributes

- [_Material](#) **Material**

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

- [CMaterial.h](#)

10.84 Math Class Reference

Class of all the static maths functions that don't fit into existing classes.

```
#include <Math.h>
```

Static Public Member Functions

- static int **random** (int min, int max)
- static XMFLOAT3 **FromScreenToWorld** (const XMFLOAT3 &vec)
Convert screen coords to world space.
- static std::string **FloatToStringWithDigits** (const float &number, const unsigned char numberOfDecimalPlaces=3, const bool preserveDecimalZeros=false, const unsigned char numberOfIntegralPlacesZeros=1)
Converts a float to a string.
- static std::string **IntToString** (const int &number, const unsigned char numberOfIntegralPlacesZeros=1)
Converts an int to a string.
- static float **DegToRad** (const float °rees)
Convert degrees to radians.
- static float **RadToDeg** (const float &radians)
Convert radians to degrees.

10.84.1 Detailed Description

Class of all the static maths functions that don't fit into existing classes.

10.84.2 Member Function Documentation

10.84.2.1 FloatToStringWithDigits()

```
std::string Math::FloatToStringWithDigits (
    const float & number,
    const unsigned char numberOfDecimalPlaces = 3,
    const bool preserveDecimalZeros = false,
    const unsigned char numberOfIntegralPlacesZeros = 1 ) [static]
```

Converts a float to a string.

Allows you to specify how many decimal places are in the string as well as zeros for both the decimal and integral parts.

Parameters

<i>number</i>	
<i>numberOfDecimalPlaces</i>	
<i>preserveDecimalZeros</i>	
<i>numberOfIntegralPlacesZeros</i>	

Returns

10.84.2.2 FromScreenToWorld()

```
XMFLOAT3 Math::FromScreenToWorld (
    const XMFLOAT3 & vec ) [static]
```

Convert screen coords to world space.

Useful for converting the mouse to world space.

Parameters

<i>vec</i>	vector to be converted to world space.
<i>camera</i>	rendering camera.

Returns

10.84.2.3 IntToString()

```
std::string Math::IntToString (
    const int & number,
    const unsigned char numberOfIntegralPlacesZeros = 1 ) [static]
```

Converts an int to a string.

Allows for extra zeros to be added infront of the string.

Parameters

<i>number</i>	
<i>numberOfIntegralPlacesZeros</i>	

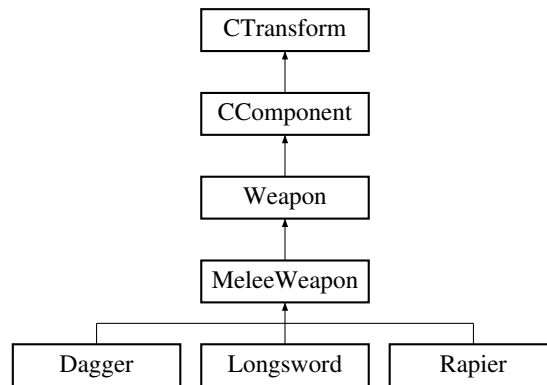
Returns

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

- [Math.h](#)
- [Math.cpp](#)

10.85 MeleeWeapon Class Reference

Inheritance diagram for MeleeWeapon:



Public Member Functions

- virtual bool [OnFire](#) ([Vector3](#) actorPos, [Vector3](#) attackDir)
Virtual OnFire function, overridden if the weapon has any unique firing logic.

Additional Inherited Members

10.85.1 Member Function Documentation

10.85.1.1 OnFire()

```
bool MeleeWeapon::OnFire (
    Vector3 actorPos,
    Vector3 attackDir ) [virtual]
```

Virtual OnFire function, overridden if the weapon has any unique firing logic.

Parameters

<i>actorPos</i>	Position of the actor using OnFire.
<i>attackDir</i>	Direction vector of the attack.

Reimplemented from [Weapon](#).

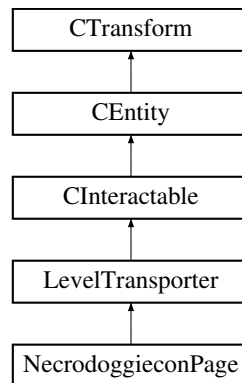
Reimplemented in [Longsword](#).

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

- [MeleeWeapon.h](#)
- [MeleeWeapon.cpp](#)

10.86 NecrodoggieconPage Class Reference

Inheritance diagram for NecrodoggieconPage:



Public Member Functions

- virtual void [OnInteract](#) () override
Called when a player has interacted with the interactable.

Protected Member Functions

- void [OnDialogueClose](#) ()

Additional Inherited Members

10.86.1 Member Function Documentation

10.86.1.1 OnInteract()

```
void NecrodoggieconPage::OnInteract ( ) [override], [virtual]
```

Called when a player has interacted with the interactable.

Reimplemented from [LevelTransporter](#).

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

- NecrodoggieconPage.h
- NecrodoggieconPage.cpp

10.87 Pathfinding Class Reference

[Pathfinding](#) class to handle all the pathfinding for the AI.

```
#include <Pathfinding.h>
```

Public Member Functions

- [Pathfinding](#) (std::vector< [CTile](#) * > waypoints)
Constructor that sets the waypoints.
- void [SetPatrolNodes](#) (std::vector< [PatrolNode](#) * > nodes)
Sets the patrol nodes and the closest waypoint to each node.
- [WaypointNode](#) * [FindClosestWaypoint](#) ([Vector3](#) position)
Finds the closest waypoint to the position passed in.
- [PatrolNode](#) * [FindClosestPatrolNode](#) ([Vector3](#) position)
Finds the closest patrol node to the position passed in.
- void [SetPath](#) ([Vector3](#) currentPosition, [WaypointNode](#) *goalWaypoint)
Gets the closest waypoint to be passed in with the goal waypoint to the calculate path function.
- void [CalculatePath](#) ([WaypointNode](#) *start, [WaypointNode](#) *goal)
A to calculate the shortest path between 2 waypoints.*
- float [CalculateCost](#) ([WaypointNode](#) *from, [WaypointNode](#) *to)
Calculates the euclidean distance between 2 waypoints.
- void [ResetNodes](#) ()
Resets the g and h costs to 10 million.
- void [DeleteNodes](#) ()
Calls the reset nodes function and clears the open, closed and path nodes arrays.
- std::vector< [WaypointNode](#) * > [GetPathNodes](#) ()
Gets the path nodes vector array.

Public Attributes

- [PatrolNode](#) * [currentPatrolNode](#)

10.87.1 Detailed Description

[Pathfinding](#) class to handle all the pathfinding for the AI.

10.87.2 Constructor & Destructor Documentation

10.87.2.1 Pathfinding()

```
Pathfinding::Pathfinding (
    std::vector< CTile * > waypoints )
```

Constructor that sets the waypoints.

Parameters

<i>waypoints</i>	Vector array of waypoints to set.
------------------	-----------------------------------

10.87.3 Member Function Documentation

10.87.3.1 CalculateCost()

```
float Pathfinding::CalculateCost (
    WaypointNode * from,
    WaypointNode * to )
```

Calculates the euclidean distance between 2 waypoints.

Parameters

<i>from</i>	Waypoint to calculate from.
<i>to</i>	Waypoint to calculate to.

Returns

Returns a float representing the distance.

10.87.3.2 CalculatePath()

```
void Pathfinding::CalculatePath (
    WaypointNode * start,
    WaypointNode * goal )
```

A* to calculate the shortest path between 2 waypoints.

Parameters

<i>start</i>	Start waypoint.
<i>goal</i>	End waypoint.

10.87.3.3 FindClosestPatrolNode()

```
PatrolNode * Pathfinding::FindClosestPatrolNode (
    Vector3 position )
```


Finds the closest patrol node to the position passed in.

Parameters

<i>position</i>	Vector3 representing the position.
-----------------	------------------------------------

Returns

Return a pointer to the closest patrol node.

10.87.3.4 FindClosestWaypoint()

```
WaypointNode * Pathfinding::FindClosestWaypoint (
    Vector3 position )
```

Finds the closest waypoint to the position passed in.

Parameters

<i>position</i>	Vector3 of the position.
-----------------	--------------------------

Returns

Returns a pointer to the closest waypoint.

10.87.3.5 GetPathNodes()

```
std::vector< WaypointNode * > Pathfinding::GetPathNodes ( )
```

Gets the path nodes vector array.

Returns

Returns the path nodes.

10.87.3.6 SetPath()

```
void Pathfinding::SetPath (
    Vector3 currentPosition,
    WaypointNode * goalWaypoint )
```

Gets the closest waypoint to be passed in with the goal waypoint to the calculate path function.

Parameters

<i>currentPosition</i>	Vector3 of the position .
<i>goalWaypoint</i>	Waypoint pointer of the goal waypoint.

10.87.3.7 SetPatrolNodes()

```
void Pathfinding::SetPatrolNodes (
    std::vector< PatrolNode * > nodes )
```

Sets the patrol nodes and the closest waypoint to each node.

Parameters

<i>nodes</i>	Vector array of patrol nodes.
--------------	-------------------------------

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

- [Pathfinding.h](#)
- [Pathfinding.cpp](#)

10.88 PatrolNode Struct Reference

Patrol node struct containing the position, closest waypoint and the next patrol node.

```
#include <CAINode.h>
```

Public Member Functions

- **PatrolNode** ([Vector3](#) pos)

Public Attributes

- [Vector3](#) **position**
- [WaypointNode](#) * **closestWaypoint**
- [PatrolNode](#) * **nextPatrolNode**

10.88.1 Detailed Description

Patrol node struct containing the position, closest waypoint and the next patrol node.

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

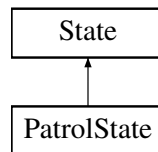
- [CAINode.h](#)

10.89 PatrolState Class Reference

[State](#) for when the AI is patrolling between the patrol points.

```
#include <State.h>
```

Inheritance diagram for PatrolState:



Public Member Functions

- void [Enter](#) ([CAIController](#) *controller) override
- void [Update](#) ([CAIController](#) *controller, float deltaTime) override
- void [Exit](#) ([CAIController](#) *controller) override

Static Public Member Functions

- static [State](#) & [getInstance](#) ()

10.89.1 Detailed Description

[State](#) for when the AI is patrolling between the patrol points.

10.89.2 Member Function Documentation

10.89.2.1 Enter()

```
void PatrolState::Enter (  
    CAIController * controller ) [override], [virtual]
```

Reimplemented from [State](#).

10.89.2.2 Exit()

```
void PatrolState::Exit (  
    CAIController * controller ) [override], [virtual]
```

Reimplemented from [State](#).

10.89.2.3 Update()

```
void PatrolState::Update (
    CAIController * controller,
    float deltaTime ) [override], [virtual]
```

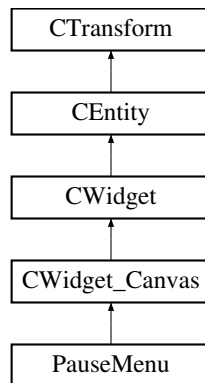
Reimplemented from [State](#).

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

- [State.h](#)
- [State.cpp](#)

10.90 PauseMenu Class Reference

Inheritance diagram for PauseMenu:



Public Member Functions

- void **PauseGame** ()
pauses game.
- void **ResumeGame** ()
resumes game.
- void **QuitToMenu** ()
returns to main menu.
- void **QuitToDesktop** ()
closes game.
- void **OpenSettingsMenu** ()
opens settings.
- virtual void [Update](#) (float deltaTime) override
listens for input to open/close pause menu through button.

Additional Inherited Members

10.90.1 Member Function Documentation

10.90.1.1 Update()

```
void PauseMenu::Update (
    float deltaTime ) [override], [virtual]
```

listens for input to open/close pause menu through button.

Parameters

<i>deltaTime</i>	
------------------	--

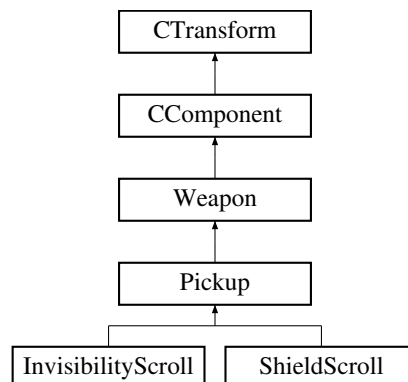
Reimplemented from [CWidget_Canvas](#).

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

- [PauseMenu.h](#)
- [PauseMenu.cpp](#)

10.91 Pickup Class Reference

Inheritance diagram for Pickup:



Public Member Functions

- void [Update](#) (float *deltaTime*) override
Update function for Cooldown of weapons.
- virtual bool [OnFire](#) ([Vector3](#) actorPos, [Vector3](#) attackDir)
Function used to try to activate the pickup.

Additional Inherited Members

10.91.1 Member Function Documentation

10.91.1.1 OnFire()

```
bool Pickup::OnFire (
    Vector3 actorPos,
    Vector3 attackDir ) [virtual]
```

Function used to try to activate the pickup.

Parameters

<i>actorPos</i>	- not used
<i>attackDir</i>	- not used

Returns

- True if it can activate, otherwise false

Reimplemented from [Weapon](#).

10.91.1.2 Update()

```
void Pickup::Update (
    float deltaTime ) [override], [virtual]
```

Update function for Cooldown of weapons.

Parameters

<i>deltaTime</i>	
------------------	--

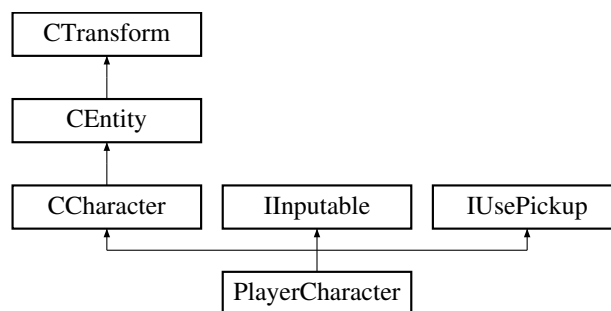
Reimplemented from [Weapon](#).

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

- Pickup.h
- [Pickup.cpp](#)

10.92 PlayerCharacter Class Reference

Inheritance diagram for PlayerCharacter:



Public Member Functions

- void [PressedHorizontal](#) (int dir, float deltaTime) override
Function inherited from interface Will use horizontal key inputs to add horizontal movement.
- void [PressedVertical](#) (int dir, float deltaTime) override
Function inherited from interface Will use vertical key inputs to add vertical movement.
- void [PressedInteract](#) () override
Function inherited from interface Will interact with objects in the world if one is available.
- void [PressedDrop](#) () override
Function inherited from interface Will drop the characters currently equipped item Will return early if the EquippedItem is null.
- void [Attack](#) () override
- void [PressedUse](#) () override
- void [UsePickup](#) (const std::string &pickupToUse, float activeTime) override
Checks the pickup item type and activates the functionality for that pickup.
- bool [GetVisible](#) ()
- virtual void [Update](#) (float deltaTime) override
Updated automatically every single frame.
- void [EquipWeapon](#) ([Weapon](#) *weapon)
- void [UpdateWeaponSprite](#) ()
- void [ApplyDamage](#) (float damage)
Public function used to apply damage to the character.
- void [ApplyDamage](#) (float damage, const std::string &onHitSound)

Public Attributes

- class [CCameraComponent](#) * **camera** = nullptr

Protected Member Functions

- void [LookAt](#) ([Vector3](#) pos)
- void [InvisibilityCallback](#) ()
Function used as a callback for when the invisibility pickup runs out.
- void [PickupTimer](#) (float deltaTime)
Function used to time how long a pickup has been active and call the appropriate callback when it runs out.
- void [ToggleVisibility](#) (bool isVisible)
Function used to toggle the visibility of the characters sprites.
- void [ToggleShield](#) (bool shield)

Protected Attributes

- float **walkSpeed** = 300
- float **walkDrag** = 10
- float **timeElapsed** = 0
- float **timeBetweenSteps** = 0.35f
- float **stepTimer**
- [CAnimationSpriteComponent](#) * **spriteComponentBody** = nullptr
- [CAnimationSpriteComponent](#) * **spriteComponentLegs** = nullptr
- [CSpriteComponent](#) * **spriteComponentShadow** = nullptr
- [CSpriteComponent](#) * **spriteComponentShield** = nullptr

- `std::vector< PlayerController * > playersController = Engine::GetEntityOfType<PlayerController>()`
- `Vector2 movementVec = { 0,0 }`
- `XMFLOAT2 movementVel = { 0,0 }`
- `XMFLOAT4 originalSpriteTint`
- `XMFLOAT4 originalLegTint`
- `const float walkAnimationSpeed = 1.3f`
- `float pickupTimer`
- `bool pickupActive`
- `float pickupActiveTime`
- `std::function< void()> pickupTimerCallback`
- `const float cameraMovementScalar = 100.0f`
- `bool hasShield = false`

10.92.1 Member Function Documentation

10.92.1.1 `ApplyDamage()` [1/2]

```
void PlayerCharacter::ApplyDamage (
    float damageAmount ) [virtual]
```

Public function used to apply damage to the character.

Reimplemented from [CCharacter](#).

10.92.1.2 `ApplyDamage()` [2/2]

```
void PlayerCharacter::ApplyDamage (
    float damage,
    const std::string & onHitSound ) [virtual]
```

Reimplemented from [CCharacter](#).

10.92.1.3 `Attack()`

```
void PlayerCharacter::Attack ( ) [override], [virtual]
```

Implements [IInputable](#).

10.92.1.4 PressedDrop()

```
void PlayerCharacter::PressedDrop ( ) [override], [virtual]
```

Function inherited from interface Will drop the characters currently equipped item Will return early if the Equipped Item is null.

Implements [IInputable](#).

10.92.1.5 PressedHorizontal()

```
void PlayerCharacter::PressedHorizontal (
    int dir,
    float deltaTime ) [override], [virtual]
```

Function inherited from interface Will use horizontal key inputs to add horizontal movement.

Parameters

<i>dir</i>	- The direction of movement, negative for left, positive for right
<i>deltaTime</i>	- Time since the last frame

Implements [IInputable](#).

10.92.1.6 PressedInteract()

```
void PlayerCharacter::PressedInteract ( ) [override], [virtual]
```

Function inherited from interface Will interact with objects in the world if one is available.

Implements [IInputable](#).

10.92.1.7 PressedUse()

```
void PlayerCharacter::PressedUse ( ) [override], [virtual]
```

Implements [IInputable](#).

10.92.1.8 PressedVertical()

```
void PlayerCharacter::PressedVertical (
    int dir,
    float deltaTime ) [override], [virtual]
```

Function inherited from interface Will use vertical key inputs to add vertical movement.

Parameters

<i>dir</i>	- The direction of movement, negative for down, positive for up
<i>deltaTime</i>	- Time since the last frame

Implements [IInputable](#).

10.92.1.9 ToggleVisibility()

```
void PlayerCharacter::ToggleVisibility (
    bool isVisible ) [protected]
```

Function used to toggle the visibility of the characters sprites.

Parameters

<i>isVisible</i>	- Whether or not the character should be visible
------------------	--

10.92.1.10 Update()

```
void PlayerCharacter::Update (
    float deltaTime ) [override], [virtual]
```

Updated automatically every single frame.

Reimplemented from [CCharacter](#).

10.92.1.11 UsePickup()

```
void PlayerCharacter::UsePickup (
    const std::string & pickupToUse,
    float activeTime ) [override], [virtual]
```

Checks the pickup item type and activates the functionality for that pickup.

E.g, Invisibility scroll will make the player invisible and bind a callback to the timer to make the player visible after a certain amount of time.

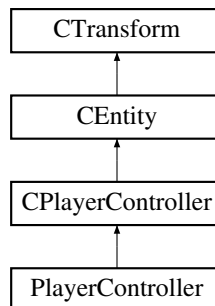
Implements [IUsePickup](#).

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

- PlayerCharacter.h
- PlayerCharacter.cpp

10.93 PlayerController Class Reference

Inheritance diagram for PlayerController:



Public Member Functions

- virtual void [Update](#) (float deltaTime) override
Inherited function Used to update the Controller each frame.

Public Attributes

- [PlayerCharacter](#) * **charOne** = nullptr

Protected Member Functions

- virtual void [HandleInput](#) (float deltaTime) override
Inherited function Used to handle the input that the Controller receives Will pass input down to the possessed Character using the [IInputable](#) interface.
- virtual void [OnPossess](#) () override
Inherited function Used to get the [IInputable](#) interface from the newly possessed character.
- virtual void [OnUnpossess](#) () override
Inherited function Used to remove the [IInputable](#) interface.
- void [OnDialogueOpen](#) ()
- void [OnDialogueClose](#) ()

Protected Attributes

- int **charIndex** = 1
- [IInputable](#) * **inputable** = nullptr
- bool **dialogueOpen** = false

10.93.1 Member Function Documentation

10.93.1.1 HandleInput()

```
void PlayerController::HandleInput (
    float deltaTime ) [override], [protected], [virtual]
```

Inherited function Used to handle the input that the Controller receives Will pass input down to the possessed Character using the [IInputable](#) interface.

Parameters

<i>deltaTime</i>	- Time since the last frame
------------------	-----------------------------

Reimplemented from [CPlayerController](#).

10.93.1.2 OnPossess()

```
void PlayerController::OnPossess ( ) [override], [protected], [virtual]
```

Inherited function Used to get the [IInputable](#) interface from the newly possessed character.

Reimplemented from [CPlayerController](#).

10.93.1.3 OnUnpossess()

```
void PlayerController::OnUnpossess ( ) [override], [protected], [virtual]
```

Inherited function Used to remove the [IInputable](#) interface.

Reimplemented from [CPlayerController](#).

10.93.1.4 Update()

```
void PlayerController::Update (
    float deltaTime ) [override], [virtual]
```

Inherited function Used to update the Controller each frame.

Parameters

<i>deltaTime</i>	- Time since the last frame
------------------	-----------------------------

Implements [CEntity](#).

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

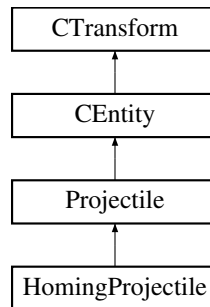
- PlayerController.h
- PlayerController.cpp

10.94 Projectile Class Reference

[Projectile](#) class for the [Projectile](#).

```
#include <Projectile.h>
```

Inheritance diagram for Projectile:



Public Member Functions

- void [StartUp](#) ([Vector3](#) dir, [Vector3](#) pos, float damage, float speed, float lifetime, int type, const std::string &projectile_name, const std::string &hitAudioPath)
Sets up the projectile based on what weapon is using it, this makes sure that the right spriate is being used.
- void [DidItHit](#) ()
Sees if the projectile is within ranged of hiting the target.
- virtual void [Update](#) (float deltaTime) override
Update for constantly moving projectile (Virtually overridden when unique logic is needed).
- void [SetLifetime](#) (float life)
- float [GetLifetime](#) ()
- [Vector3](#) [GetPosition](#) ()
- void [SetPosition](#) ([Vector3](#) newPosition)
- [Vector3](#) [GetDirection](#) ()
- float [GetSpeed](#) ()
- USERTYPE2 [GetUserType](#) ()

Public Attributes

- class [CSpriteComponent](#) * [ProjectileSprite](#) = nullptr

Additional Inherited Members

10.94.1 Detailed Description

[Projectile](#) class for the [Projectile](#).

10.94.2 Member Function Documentation

10.94.2.1 DidItHit()

```
void Projectile::DidItHit ( )
```

Sees if the projectile is within ranged of hiting the target.

\Damages the target if it hit

10.94.2.2 StartUp()

```
void Projectile::StartUp (
    Vector3 dir,
    Vector3 pos,
    float damage,
    float speed,
    float lifetime,
    int type,
    const std::string & projectile_name,
    const std::string & hitAudioPath )
```

Sets up the projectile based on what weapon is using it, this makes sure that the right spriate is being used.

This also allows for the projectile to be at the right rotation when firing

10.94.2.3 Update()

```
void Projectile::Update (
    float deltaTime ) [override], [virtual]
```

Update for constantly moving projectile (Virtually overridden when unique logic is needed).

Parameters

<i>deltaTime</i>	
------------------	--

Implements [CEntity](#).

Reimplemented in [HomingProjectile](#).

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

- [Projectile.h](#)
- [Projectile.cpp](#)

10.95 PropData Struct Reference

Public Attributes

- std::string **propName**

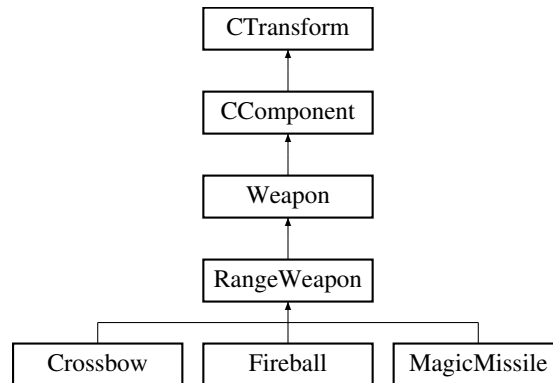
- [Vector2](#) collisionData
- [Vector2](#) atlasSize

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

- CWorld_Edit.h

10.96 RangeWeapon Class Reference

Inheritance diagram for RangeWeapon:



Public Member Functions

- virtual bool **OnFire** ([Vector3](#) actorPos, [Vector3](#) attackDir)
Sees if there is any ammo in the weapon if there is then it will fire it.
- void **SetProjectileSpeed** (float speed)
- float **GetProjectileSpeed** ()

Additional Inherited Members

10.96.1 Member Function Documentation

10.96.1.1 OnFire()

```

bool RangeWeapon::OnFire (
    Vector3 actorPos,
    Vector3 attackDir ) [virtual]
  
```

Sees if there is any ammo in the weapon if there is then it will fire it.

\Gets the weapon system ready to make the projectile

Reimplemented from [Weapon](#).

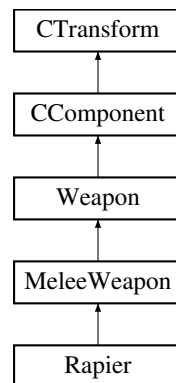
Reimplemented in [MagicMissile](#).

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

- RangeWeapon.h
- RangeWeapon.cpp

10.97 Rapier Class Reference

Inheritance diagram for Rapier:



Additional Inherited Members

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

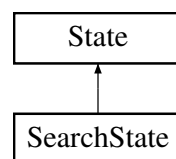
- [Rapier.h](#)
- [Rapier.cpp](#)

10.98 SearchState Class Reference

[State](#) for when the AI is searching for the player.

```
#include <State.h>
```

Inheritance diagram for SearchState:



Public Member Functions

- void [Enter](#) ([CAIController](#) *controller) override
- void [Update](#) ([CAIController](#) *controller, float deltaTime) override
- void [Exit](#) ([CAIController](#) *controller) override

Static Public Member Functions

- static [State](#) & [getInstance](#) ()

10.98.1 Detailed Description

[State](#) for when the AI is searching for the player.

The AI will spin on the spot looking for the player.

10.98.2 Member Function Documentation

10.98.2.1 Enter()

```
void SearchState::Enter (  
    CAIController * controller ) [override], [virtual]
```

Reimplemented from [State](#).

10.98.2.2 Exit()

```
void SearchState::Exit (  
    CAIController * controller ) [override], [virtual]
```

Reimplemented from [State](#).

10.98.2.3 Update()

```
void SearchState::Update (  
    CAIController * controller,  
    float deltaTime ) [override], [virtual]
```

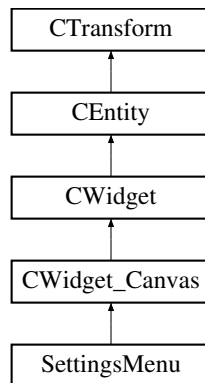
Reimplemented from [State](#).

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

- [State.h](#)
- [State.cpp](#)

10.99 SettingsMenu Class Reference

Inheritance diagram for SettingsMenu:



Public Member Functions

- void **CloseSettings** ()
closes settings and re-opens either main menu or pause menu depending on which is applicable.
- virtual void **Update** (float deltaTime) override
Updated automatically every single frame.

Additional Inherited Members

10.99.1 Member Function Documentation

10.99.1.1 Update()

```
void SettingsMenu::Update (  
    float deltaTime ) [override], [virtual]
```

Updated automatically every single frame.

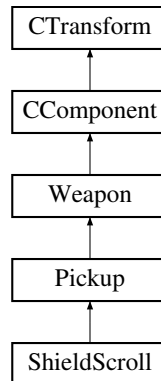
Reimplemented from [CWidget_Canvas](#).

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

- [SettingsMenu.h](#)
- [SettingsMenu.cpp](#)

10.100 ShieldScroll Class Reference

Inheritance diagram for ShieldScroll:



Additional Inherited Members

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

- ShieldScroll.h
- ShieldScroll.cpp

10.101 SimpleVertex Struct Reference

Public Attributes

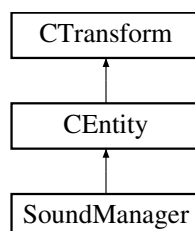
- XMFLOAT3 **Pos**
- XMFLOAT2 **TexCoord**

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

- [CMesh.h](#)

10.102 SoundManager Class Reference

Inheritance diagram for SoundManager:



Static Public Member Functions

- static void **Initialise** ()
Function to initialise the [SoundManager](#) by creating audio emitters for each sound that will be used within the game.
- static void **AddSound** (const std::string &audioPath, const std::string &audioName, float audioRange)
Function to add a new audio emitter to the [SoundManager](#).
- static void **AddSound** (const std::string &audioPath, const std::string &audioName, float audioRange, bool ambient)
Function to add a new audio emitter to the [SoundManager](#).
- static void **PlaySound** (const std::string &audioName, [Vector3](#) position)
Function to play audio from one of the audio emitters stored in the [SoundManager](#).
- static void **PlayMusic** (const std::string &musicPath, [CEntity](#) *attachedEntity)
Function used to play music.

Additional Inherited Members

10.102.1 Member Function Documentation

10.102.1.1 AddSound() [1/2]

```
void SoundManager::AddSound (
    const std::string & audioPath,
    const std::string & audioName,
    float audioRange ) [static]
```

Function to add a new audio emitter to the [SoundManager](#).

Parameters

<i>audioPath</i>	- Path to the audio file
<i>audioName</i>	- Name to store in the map with the emitter
<i>audioRange</i>	- The range of the audio

10.102.1.2 AddSound() [2/2]

```
void SoundManager::AddSound (
    const std::string & audioPath,
    const std::string & audioName,
    float audioRange,
    bool ambient ) [static]
```

Function to add a new audio emitter to the [SoundManager](#).

Parameters

<i>audioPath</i>	- Path to the audio file
<i>audioName</i>	- Name to store in the map with the emitter
<i>audioRange</i>	- The range of the audio
<i>ambient</i>	- Whether the audio should be ambient or not

10.102.1.3 PlayMusic()

```
void SoundManager::PlayMusic (
    const std::string & musicPath,
    CEntity * attachedEntity ) [static]
```

Function used to play music.

Parameters

<i>musicPath</i>	- Path to the audio file containing the correct music
<i>attachedEntity</i>	- The entity that the music should follow to ensure it can always be heard

10.102.1.4 PlaySound()

```
void SoundManager::PlaySound (
    const std::string & audioName,
    Vector3 position ) [static]
```

Function to play audio from one of the audio emitters stored in the [SoundManager](#).

Parameters

<i>audioName</i>	- The name associated with the audio emitter
<i>position</i>	- The position to play the audio at

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

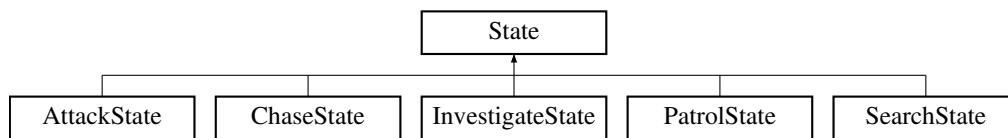
- SoundManager.h
- [SoundManager.cpp](#)

10.103 State Class Reference

Base state class.

```
#include <State.h>
```

Inheritance diagram for State:



Public Member Functions

- virtual void **Enter** ([CAIController](#) *controller)
- virtual void **Exit** ([CAIController](#) *controller)
- virtual void **Update** ([CAIController](#) *controller, float deltaTime)

10.103.1 Detailed Description

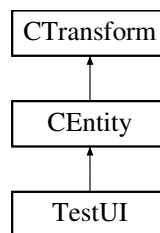
Base state class.

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

- [State.h](#)

10.104 TestUI Class Reference

Inheritance diagram for TestUI:



Public Member Functions

- virtual void [Update](#) (float deltaTime) override
Updated automatically every single frame.

Additional Inherited Members

10.104.1 Member Function Documentation

10.104.1.1 Update()

```
void TestUI::Update (
    float deltaTime ) [override], [virtual]
```

Updated automatically every single frame.

Implements [CEntity](#).

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

- TestUI.h
- TestUI.cpp

10.105 Vector2Base< T > Class Template Reference

Public Member Functions

- **Vector2Base** (DirectX::XMFLOAT3 Input)
- **Vector2Base** (T X, T Y)
- **Vector2Base** (T AllAxis)
- **Vector2Base** (__m128 Data)
- DirectX::XMFLOAT3 **ToXMFLOAT3** ()
- **Vector2Base operator*** (const T &OtherFloat) const
- **Vector2Base operator/** (const T &OtherFloat) const
- **Vector2Base operator+** (const T &OtherFloat) const
- **Vector2Base operator-** (const T &OtherFloat) const
- **Vector2Base operator*** (const **Vector2Base** OtherVector) const
- **Vector2Base operator-** (const **Vector2Base** OtherVector) const
- **Vector2Base operator+** (const **Vector2Base** OtherVector) const
- **Vector2Base operator/** (const **Vector2Base** OtherVector) const
- **Vector2Base & operator+=** (const **Vector2Base** &OtherVector)
- **Vector2Base & operator*=** (const **Vector2Base** &OtherVector)
- **Vector2Base & operator/=** (const **Vector2Base** &OtherVector)
- **Vector2Base & operator-=** (const **Vector2Base** &OtherVector)
- bool **operator==** (const **Vector2Base** &B) const
- bool **operator!=** (const **Vector2Base** &B) const
- float **Magnitude** () const
- float **Dot** (const **Vector2Base** OtherVector) const
- float **DistanceTo** (const **Vector2Base** B)
- **Vector2Base & Normalize** ()
- float **Determinant** (const **Vector2Base** OtherVector)
- **Vector2Base Lerp** (const **Vector2Base** A, const **Vector2Base** B, float Alpha)
- void **Truncate** (float max)

Public Attributes

- ```

union {
 struct {
 T x
 T y
 }
 __m128 intrinsic
};

```

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

- Vector3.h

## 10.106 Vector3Base< T > Class Template Reference

### Public Member Functions

- **Vector3Base** (DirectX::XMFLOAT3 Input)
- **Vector3Base** (T X, T Y, T Z)
- **Vector3Base** (T AllAxis)
- **Vector3Base** (\_\_m128 Data)
- DirectX::XMFLOAT3 **ToXMFLOAT3** ()
- **Vector3Base** **operator\*** (const T &OtherFloat) const
- **Vector3Base** **operator/** (const T &OtherFloat) const
- **Vector3Base** **operator+** (const T &OtherFloat) const
- **Vector3Base** **operator-** (const T &OtherFloat) const
- **Vector3Base** **operator\*** (const **Vector3Base** OtherVector) const
- **Vector3Base** **operator-** (const **Vector3Base** OtherVector) const
- **Vector3Base** **operator+** (const **Vector3Base** OtherVector) const
- **Vector3Base** **operator/** (const **Vector3Base** OtherVector) const
- **Vector3Base** & **operator+=** (const **Vector3Base** &OtherVector)
- **Vector3Base** & **operator\*=** (const **Vector3Base** &OtherVector)
- **Vector3Base** & **operator/=** (const **Vector3Base** &OtherVector)
- **Vector3Base** & **operator-=** (const **Vector3Base** &OtherVector)
- bool **operator==** (const **Vector3Base** &B) const
- bool **operator!=** (const **Vector3Base** &B) const
- float **Magnitude** () const
- float **Dot** (const **Vector3Base** OtherVector) const
- float **DistanceTo** (const **Vector3Base** B)
- **Vector3Base** & **Normalize** ()
- float **Determinant** (const **Vector3Base** OtherVector)
- **Vector3Base** **Lerp** (const **Vector3Base** A, const **Vector3Base** B, float Alpha)
- void **Truncate** (float max)

## Public Attributes

- ```
union {  
    struct {  
        T x  
        T y  
        T z  
    }  
    __m128 intrinsic  
};
```

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

- [Vector3.h](#)

10.107 WaypointNode Struct Reference

Waypoint node struct containing the waypoint, parent waypoint, neighbours and the costs.

```
#include <CAINode.h>
```

Public Attributes

- [CTile](#) * **waypoint** = nullptr
- [CTile](#) * **parentWaypoint** = nullptr
- std::vector< [WaypointNode](#) * > **neighbours**
- float **gCost** = 0.0f
- float **hCost** = 0.0f
- float **fCost** = 0.0f

10.107.1 Detailed Description

Waypoint node struct containing the waypoint, parent waypoint, neighbours and the costs.

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

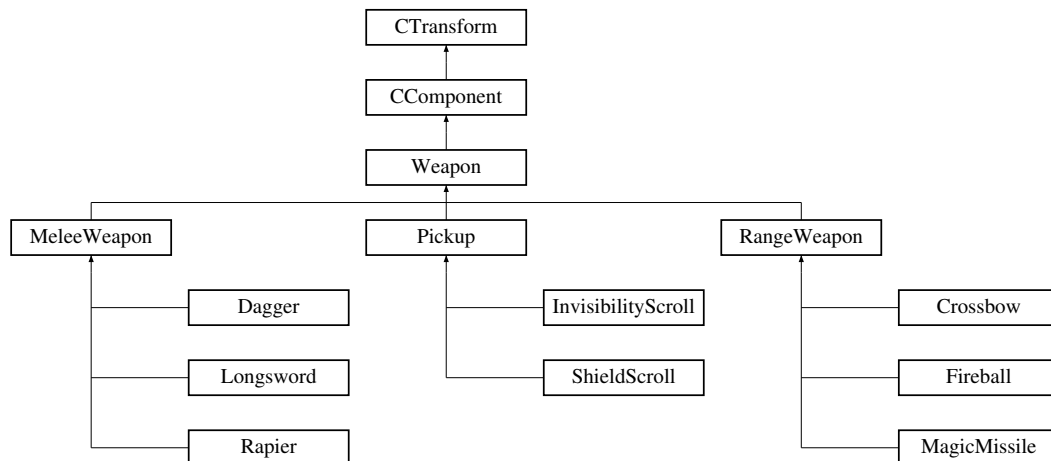
- [CAINode.h](#)

10.108 Weapon Class Reference

Base [Weapon](#) class inherited by all weapons.

```
#include <weapons.h>
```

Inheritance diagram for [Weapon](#):



Public Member Functions

- **Weapon** (std::string weapon="Dagger")
- virtual bool **OnFire** ([Vector3](#) actorPos, [Vector3](#) attackDir)
OnFire function of base [Weapon](#) class, this is overridden in the [MeleeWeapon](#) and [RangeWeapon](#) sub-classes.
- void **SetWeapon** (int ID)
Sets the private variables using the information stored in a JSON file of weapons.
- void **SetWeapon** (std::string ID)
- std::string **IDToName** (int ID)
- int **NameToID** (std::string Name)
- virtual void **Update** (float deltaTime) override
Update function for Cooldown of weapons.
- virtual void **Draw** ([ID3D11DeviceContext](#) *context, const [XMFLOAT4X4](#) &parentMat, [ConstantBuffer](#) cb, [ID3D11Buffer](#) *constantBuffer) override
Almost the same as [Update\(\)](#) but to be used for drawing only.
- void **SetUserType** ([USER_TYPE](#) userType)
- std::string **GetType** ()
- std::string **GetProjectileIcon** ()
- float **GetDamage** ()
- float **GetRange** ()
- float **GetAttack_Speed** ()
- float **GetMaxAmmo** ()
- void **SetMaxAmmo** (float amount)
- float **GetAmmo** ()
- void **SetAmmo** (float amount)
- bool **GetUnique** ()
- bool **GetCanFire** ()
- void **SetCanFire** (bool canFire)
- void **SetTextureOffset** ([XMFLOAT2](#) offset)

- XMFLOAT2 **GetTextureOffset** ()
- void **SetRenderRect** (XMUINT2 rect)
- XMUINT2 **GetRenderRect** ()
- void **SetScale** (XMFLOAT3 setScale)
- XMFLOAT3 **GetScale** ()
- USERTYPE **GetUserType** ()
- std::string **GetName** ()
- std::string **GetIconPath** ()
- std::string **GetHitSound** ()
- std::string **GetAttackSound** ()
- void **StartCooldown** ()

Protected Attributes

- std::string **pickupType**

10.108.1 Detailed Description

Base [Weapon](#) class inherited by all weapons.

10.108.2 Member Function Documentation

10.108.2.1 Draw()

```
void Weapon::Draw (  
    ID3D11DeviceContext * context,  
    const XMFL0AT4X4 & parentMat,  
    ConstantBuffer cb,  
    ID3D11Buffer * constantBuffer ) [override], [virtual]
```

Almost the same as [Update\(\)](#) but to be used for drawing only.

Implements [CComponent](#).

10.108.2.2 OnFire()

```
bool Weapon::OnFire (  
    Vector3 actorPos,  
    Vector3 attackDir ) [virtual]
```

OnFire function of base [Weapon](#) class, this is overridden in the [MeleeWeapon](#) and [RangeWeapon](#) sub-classes.

Parameters

<i>actorPos</i>	Position of the actor that is using the function (Used for virtual overriding)
<i>attackDir</i>	Direction of the attack (Used for virtual overriding)

Reimplemented in [Longsword](#), [MeleeWeapon](#), [Pickup](#), [MagicMissile](#), and [RangeWeapon](#).

10.108.2.3 SetWeapon()

```
void Weapon::SetWeapon (
    int ID )
```

Sets the private variables using the information stored in a JSON file of weapons.

Parameters

<i>weapon</i>	Name of the weapon in the JSON
---------------	--------------------------------

10.108.2.4 Update()

```
void Weapon::Update (
    float deltaTime ) [override], [virtual]
```

Update function for Cooldown of weapons.

Parameters

<i>deltaTime</i>	
------------------	--

Implements [CComponent](#).

Reimplemented in [Crossbow](#), and [Pickup](#).

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

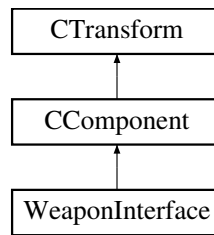
- [weapons.h](#)
- [weapons.cpp](#)

10.109 WeaponInterface Class Reference

[Weapon](#) Interface class used to switch weapons being used through the Strategy Design Pattern.

```
#include <WeaponInterface.h>
```

Inheritance diagram for WeaponInterface:



Public Member Functions

- virtual bool [OnFire](#) ([Vector3](#) actorPos, [Vector3](#) attackDir)
OnFire Function calls CurrentWeapon OnFire, this OnFire is overridden through virtual functions in the Sub-Classes.
- virtual void [Update](#) (float deltaTime) override
Updated automatically every single frame.
- virtual void [Draw](#) (ID3D11DeviceContext *context, const XMFLOAT4X4 &parentMat, [ConstantBuffer](#) cb, ID3D11Buffer *constantBuffer) override
Almost the same as [Update\(\)](#) but to be used for drawing only.
- void [SetWeapon](#) ([Weapon](#) *weapon)
Function to delete previous weapon from memory and set in use weapon.
- [Weapon](#) * [GetCurrentWeapon](#) ()
- void [SetUserType](#) (USER_TYPE userType)
Sets type of user using the weapon.
- USER_TYPE [GetUserType](#) ()

Additional Inherited Members

10.109.1 Detailed Description

[Weapon](#) Interface class used to switch weapons being used through the Strategy Design Pattern.

10.109.2 Member Function Documentation

10.109.2.1 Draw()

```

void WeaponInterface::Draw (
    ID3D11DeviceContext * context,
    const XMFLOAT4X4 & parentMat,
    ConstantBuffer cb,
    ID3D11Buffer * constantBuffer ) [override], [virtual]
  
```

Almost the same as [Update\(\)](#) but to be used for drawing only.

Implements [CComponent](#).

10.109.2.2 OnFire()

```
bool WeaponInterface::OnFire (
    Vector3 actorPos,
    Vector3 attackDir ) [virtual]
```

OnFire Function calls CurrentWeapon OnFire, this OnFire is overridden through virtual functions in the Sub-Classes.

Parameters

<i>actorPos</i>	
<i>attackDir</i>	

10.109.2.3 SetUserType()

```
void WeaponInterface::SetUserType (
    USERTYPE userType )
```

Sets type of user using the weapon.

Parameters

<i>userType</i>	
-----------------	--

10.109.2.4 SetWeapon()

```
void WeaponInterface::SetWeapon (
    Weapon * weapon )
```

Function to delete previous weapon from memory and set in use weapon.

Parameters

<i>weapon</i>	
---------------	--

10.109.2.5 Update()

```
void WeaponInterface::Update (
    float deltaTime ) [override], [virtual]
```

Updated automatically every single frame.

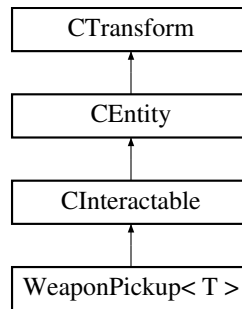
Implements [CComponent](#).

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

- [WeaponInterface.h](#)
- [WeaponInterface.cpp](#)

10.110 WeaponPickup< T > Class Template Reference

Inheritance diagram for WeaponPickup< T >:



Public Member Functions

- virtual void [OnInteract](#) () override
Updates the player character's weapon when the player interacts.
- void [SetWeapon](#) (T *weapon)
Sets the weapon of the pickup.

Additional Inherited Members

10.110.1 Member Function Documentation

10.110.1.1 OnInteract()

```
template<typename T >
void WeaponPickup< T >::OnInteract [inline], [override], [virtual]
```

Updates the player character's weapon when the player interacts.

Reimplemented from [CInteractable](#).

10.110.1.2 SetWeapon()

```
template<typename T >
void WeaponPickup< T >::SetWeapon (
    T * weapon ) [inline]
```

Sets the weapon of the pickup.

Parameters

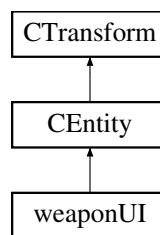
<i>weapon</i>	weapon that the pickup will be set to.
---------------	--

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

- [WeaponPickup.h](#)

10.111 weaponUI Class Reference

Inheritance diagram for weaponUI:



Public Member Functions

- **weaponUI ()**
Sets up all of the UI elements.
- virtual void **updateUI** (std::string WeaponName, int currentAmmo, int maxAmmo, std::string spritePath)
Updates [Weapon](#) UI elements when called ideally after a change is made.
- virtual void **Update** (float deltaTime) override
Updates timer each frame.

Additional Inherited Members

10.111.1 Member Function Documentation

10.111.1.1 Update()

```
void weaponUI::Update (
    float deltaTime ) [override], [virtual]
```

Updates timer each frame.

Parameters

<i>deltaTime</i>	
------------------	--

Implements [CEntity](#).

10.111.1.2 updateUI()

```
void weaponUI::updateUI (
    std::string weaponName,
    int currentAmmo,
    int maxAmmo,
    std::string spritePath ) [virtual]
```

Updates [Weapon](#) UI elements when called ideally after a change is made.

Parameters

<i>weaponName</i>	
<i>currentAmmo</i>	
<i>maxAmmo</i>	
<i>spritePath</i>	

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

- [weaponUI.h](#)
- [weaponUI.cpp](#)

Chapter 11

File Documentation

11.1 CAINode.h File Reference

Header containing all the nodes used by the AI.

```
#include "Cerberus/Core/Utility/Vector3.h"
#include <iostream>
#include <vector>
```

Classes

- struct [WaypointNode](#)
Waypoint node struct containing the waypoint, parent waypoint, neighbours and the costs.
- struct [PatrolNode](#)
Patrol node struct containing the position, closest waypoint and the next patrol node.

11.1.1 Detailed Description

Header containing all the nodes used by the AI.

Author

Nasser Ksous

Date

May 2022

11.2 CAINode.h

[Go to the documentation of this file.](#)

```
1 #pragma once
2 /*****
9 class CTile;
10 #include "Cerberus/Core/Utility/Vector3.h"
11 #include <iostream>
12 #include <vector>
16 struct WaypointNode
17 {
18     CTile* waypoint = nullptr;
19     CTile* parentWaypoint = nullptr;
20     std::vector<WaypointNode*> neighbours;
21     float gCost = 0.0f;
22     float hCost = 0.0f;
23     float fCost = 0.0f;
24 };
25
29 struct PatrolNode
30 {
31     Vector3 position;
32     WaypointNode* closestWaypoint;
33     PatrolNode* nextPatrolNode;
34
35     PatrolNode(Vector3 pos) : position(pos)
36     {
37         closestWaypoint = nullptr;
38         nextPatrolNode = nullptr;
39     }
40 };
```

11.3 Pathfinding.cpp File Reference

All the necessary functions to help any AI to traverse any level.

```
#include "Pathfinding.h"
#include "Cerberus/Core/Environment/CTile.h"
```

11.3.1 Detailed Description

All the necessary functions to help any AI to traverse any level.

Author

Nasser Ksous

Date

May 2022

11.4 Pathfinding.h File Reference

Class that handles all the necessary functions and variables for the AI to navigate through any level.

```
#include "CAINode.h"
```

Classes

- class [Pathfinding](#)
Pathfinding class to handle all the pathfinding for the AI.

11.4.1 Detailed Description

Class that handles all the necessary functions and variables for the AI to navigate through any level.

Author

Nasser Ksous

Date

May 2022

11.5 Pathfinding.h

[Go to the documentation of this file.](#)

```
1 #pragma once
2 /*****
9 #include "CAINode.h"
10
14 class Pathfinding
15 {
16 public:
17     Pathfinding(std::vector<CTile*> waypoints);
18     ~Pathfinding();
19
20     void SetPatrolNodes(std::vector<PatrolNode*> nodes);
21     WaypointNode* FindClosestWaypoint(Vector3 position);
22     PatrolNode* FindClosestPatrolNode(Vector3 position);
23
24     void SetPath(Vector3 currentPosition, WaypointNode* goalWaypoint);
25     void CalculatePath(WaypointNode* start, WaypointNode* goal);
26     float CalculateCost(WaypointNode* from, WaypointNode* to);
27     void ResetNodes();
28     void DeleteNodes();
29
30     std::vector<WaypointNode*> GetPathNodes();
31
32     PatrolNode* currentPatrolNode;
33
34 private:
35     std::vector<WaypointNode*> open;
36     std::vector<WaypointNode*> closed;
37     std::vector<WaypointNode*> waypointNodes;
38     // Array of nodes on the path from goal to start.
39     std::vector<WaypointNode*> pathNodes;
40     std::vector<PatrolNode*> patrolNodes;
41 };
42
```

11.6 CComponent.h File Reference

Fundamental component class of the engine.

```
#include "Cerberus\Core\Engine.h"
#include "Cerberus\Core\Utility\Vector3.h"
#include "Cerberus/Core/Utility/CTransform.h"
```

Classes

- class [CComponent](#)

Fundamental component class of the engine.

11.6.1 Detailed Description

Fundamental component class of the engine.

Author

Arrien Bidmead

Date

January 2022

11.7 CComponent.h

[Go to the documentation of this file.](#)

```
1  /*****
9  #pragma once
10 #include "Cerberus\Core\Engine.h"
11 #include "Cerberus\Core\Utility\Vector3.h"
12 #include "Cerberus\Core\Utility\CTransform.h"
13
18 class CComponent : public CTransform
19 {
20     XMFLOAT2 anchor = { 0.5,0.5 };
21     XMUINT2 lastResolution = { 0,0 };
22
23     class CEntity* parent = nullptr;
24
25     bool translucency = false;
26
27     bool ui = false;
28
29     bool shouldUpdate = true;
30     bool shouldDraw = false;
31
32     std::string name = "UNNAMED COMPONENT";
33
34 public:
40     void SetAnchor(const XMFLOAT2& newAnchor) { anchor = newAnchor; updateTransform = true; }
41
49     virtual void SetUseTranslucency(const bool& newTranslucency);
50
54     void SetIsUI(const bool& newIsUI) { ui = newIsUI; }
55
59     void SetShouldUpdate(const bool& newShouldUpdate) { shouldUpdate = newShouldUpdate; }
60
64     void SetShouldDraw(const bool& newShouldDraw) { shouldDraw = newShouldDraw; }
65
69     void SetLastResolution(const XMUINT2& newLastResolution) { lastResolution = newLastResolution; }
70
74     void SetParent(class CEntity* newParent);
75
79     void SetName(const std::string& newName) { name = newName.c_str(); }
80
81     const bool& GetShouldUpdate() const { return shouldUpdate; }
82     const bool& GetShouldDraw() const { return shouldDraw; }
83     const bool& GetIsUI() const { return ui; }
84     const XMUINT2& GetLastResolution() const { return lastResolution; }
85     const bool& GetUseTranslucency() const { return translucency; }
86     const XMFLOAT2& GetAnchor() const { return anchor; }
87     class CEntity* GetParent() const { return parent; }
88     const std::string& GetName() const { return name; }
89     const std::string GetDebugInfo() const;
90
94     XMFLOAT3 GetWorldPosition();
95     virtual XMFLOAT4X4 GetTransform() override;
96
100     virtual void Update(float deltaTime) = 0;
101
105     virtual void Draw(struct ID3D11DeviceContext* context, const XMFLOAT4X4& parentMat, ConstantBuffer
cb, ID3D11Buffer* constantBuffer) = 0;
106     virtual ~CComponent() {};
107 };
```


11.8 CEntity.h File Reference

Fundamental class of the engine with a world transform and ability to have components.

```
#include "Cerberus\Core\CComponent.h"
#include "Cerberus/Core/Utility/CollisionManager/CollisionComponent.h"
#include "Cerberus\Core\Utility\Vector3.h"
```

Classes

- class [CEntity](#)

Fundamental class of the engine with a world transform and ability to have components.

11.8.1 Detailed Description

Fundamental class of the engine with a world transform and ability to have components.

Author

Arrien Bidmead

Date

January 2022

11.9 CEntity.h

[Go to the documentation of this file.](#)

```
1  /*****
9  #pragma once
10
11 #include "Cerberus\Core\CComponent.h"
12 #include "Cerberus/Core/Utility/CollisionManager/CollisionComponent.h"
13 #include "Cerberus\Core\Utility\Vector3.h"
14
19 class CEntity : public CTransform
20 {
21     bool shouldUpdate = true;
22     bool shouldMove = false;
23     bool visible = true;
24     bool ui = false;
25
26     std::vector<CComponent*> components;
27
28 public:
32     void SetShouldUpdate(const bool& newShouldUpdate) { shouldUpdate = newShouldUpdate; }
33
37     void SetShouldMove(const bool& newShouldMove) { shouldMove = newShouldMove; }
38
42     void SetVisible(const bool& newVisibility) { visible = newVisibility; }
43
48     void SetIsUI(const bool& newUI) { ui = newUI; }
49
50     const bool& GetShouldUpdate() const { return shouldUpdate; }
51     const bool& GetShouldMove() const { return shouldMove; }
52     const bool& GetVisible() const { return visible; }
53     const bool& GetIsUI() const { return ui; }
54     const std::vector<CComponent*>& GetAllComponents() const { return components; }
55
59     virtual void Update(float deltaTime) = 0;
60     virtual ~CEntity();
```

```

61
62     template <class T>
63     T* AddComponent(const std::string& componentName)
64     {
65         CComponent* tmp = new T();
66         tmp->SetParent(this);
67         tmp->SetName(componentName);
68         components.push_back(tmp);
69         EntityManager::AddComponent(tmp);
70         return dynamic_cast<T*>(tmp);
71     }
72
73     template<class T>
74     T* GetComponentOfType()
75     {
76         T* comp = nullptr;
77         for(auto& component : components)
78         {
79             comp = dynamic_cast<T*>(component);
80             if(comp != nullptr)
81             {
82                 return comp;
83             }
84         }
85
86         return nullptr;
87     }
88
89     template<class T>
90     std::vector<T*> GetAllComponentsOfType()
91     {
92         std::vector<T*> output;
93         T* comp = nullptr;
94         for (auto& component : components)
95         {
96             comp = dynamic_cast<T*>(component);
97             if (comp != nullptr)
98             {
99                 output.push_back(comp);
100             }
101         }
102
103         return output;
104     }
105
106     void RemoveComponent(CComponent* reference);
107
108     CollisionComponent* colComponent = nullptr;
109     virtual void HasCollided(CollisionComponent* collidedObject)
110     {
111         if (!collidedObject->GetTrigger())
112         {
113             if (collidedObject->GetName() != "Enemy")
114             {
115                 colComponent->Resolve(collidedObject);
116                 this->SetPosition(colComponent->GetPosition());
117             }
118         }
119     };
120
121 };
122
123 };

```

11.10 CAnimationSpriteComponent.h File Reference

Extends [CSpriteComponent](#) to automatically animate sprite sheets.

```
#include "CSpriteComponent.h"
```

Classes

- class [CAnimationSpriteComponent](#)

Extends [CSpriteComponent](#) to automatically animate sprite-sheets.

11.10.1 Detailed Description

Extends [CSpriteComponent](#) to automatically animate sprite sheets.

This class will automatically animate a region of a sprite-sheet. Its up to you to input the region of the sprite-sheet to animate.

Author

Arrien Bidmead

Date

May 2022

11.11 CAnimationSpriteComponent.h

[Go to the documentation of this file.](#)

```
1  /*****
12 #pragma once
13 #include "CSpriteComponent.h"
14
15 class CAnimationSpriteComponent : public CSpriteComponent
16 {
17     float timeElapsed = 0.0f;
18     float animSpeed = 24.0f;
19     bool playing = true;
20     XMUINT2 animationRectSize = { 1,1 };
21     XMUINT2 animationRectPosition = { 0,0 };
22     XMUINT2 currentFrame = { 0,0 }; //relative to the animation rect.
23
24 public:
25     void ResetAnimation();
26
27     void SetAnimationRectSize(const XMUINT2& newSize, const bool& resetAnimation = false) {
28         animationRectSize = newSize; if (resetAnimation) ResetAnimation(); };
29     const XMUINT2& GetAnimationRectSize() { return animationRectSize; };
30
31     void SetAnimationRectPosition(const XMUINT2& newPosition, const bool& resetAnimation = false) {
32         animationRectPosition = newPosition; if (resetAnimation) ResetAnimation(); };
33     const XMUINT2& GetAnimationRectPosition() { return animationRectPosition; };
34
35     const XMUINT2& GetCurrentFrame() { return currentFrame; };
36
37     void SetPlaying(const bool& newState, const bool& resetAnimation = false) { playing = newState; if
38         (resetAnimation) ResetAnimation(); };
39     const bool& GetPlaying() { return playing; };
40
41     void SetElapsedTime(const float& newTime) { timeElapsed = newTime; };
42     const float& GetElapsedTime() { return timeElapsed; };
43
44     void SetAnimationSpeed(const float& newSpeed) { animSpeed = newSpeed; };
45     const float& GetAnimationSpeed() { return animSpeed; };
46
47     CAnimationSpriteComponent();
48     virtual void Update(float deltaTime) override;
49 };
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
25
```

Classes

- class [CAudioEmitterComponent](#)

11.12.1 Detailed Description

Allows a entity to emit audio.

Author

Luke Whiting

Date

Jan 2021

11.13 CAudioEmitterComponent.h

[Go to the documentation of this file.](#)

```
1  /*****
8  #pragma once
9  #include "Cerberus\Core\CComponent.h"
10 #include "Cerberus/Core/Utility/Audio/AudioController.h"
11 #include "Cerberus/Core/Utility/DebugOutput/Debug.h"
12
13 //Fundimental component class
14 //Can be extended upon to make new components to add to CEntity
15 class CAudioEmitterComponent : public CComponent
16 {
17 public:
18     CAudioEmitterComponent();
19     ~CAudioEmitterComponent();
20     void Load(const std::string& path);
21     void Load(const std::string& path, bool ambient);
22     void Play();
23     void Play(bool loop);
24     void Stop();
25     void SetRange(float range);
26
27     //Updated automatically every single frame
28     virtual void Update(float deltaTime);
29
30     virtual void Draw(struct ID3D11DeviceContext* context, const XMFL0AT4X4& parentMat, ConstantBuffer
31         cb, ID3D11Buffer* constantBuffer)
32     {
33         UNREFERENCED_PARAMETER(context);
34         UNREFERENCED_PARAMETER(parentMat);
35         UNREFERENCED_PARAMETER(cb);
36         UNREFERENCED_PARAMETER(constantBuffer);
37     };
38 private:
39     CEmitter* emitter;
40 };
```

11.14 CCameraComponent.h File Reference

Used to attach a camera to a entity.

```
#include <DirectXMath.h>
#include "Cerberus/Core/CComponent.h"
#include "Cerberus/Core/CEntity.h"
```

Classes

- class [CCameraComponent](#)

11.14.1 Detailed Description

Used to attach a camera to a entity.

Author

Luke Whiting

Date

May 2022

11.15 CCameraComponent.h

[Go to the documentation of this file.](#)

```
1  /*****
9  #pragma once
10 #include <DirectXMath.h>
11 #include "Cerberus/Core/CComponent.h"
12 #include "Cerberus/Core/CEntity.h"
13 class CCameraComponent : public CComponent
14 {
15 public:
16     CCameraComponent();
17     virtual ~CCameraComponent();
18
19     virtual void Update(float deltaTime) override;
20     virtual void Draw(struct ID3D11DeviceContext* context, const XMFLOAT4X4& parentMat, ConstantBuffer
        cb, ID3D11Buffer* constantBuffer) override {UNREFERENCED_PARAMETER(context);
        UNREFERENCED_PARAMETER(parentMat); UNREFERENCED_PARAMETER(cb);
        UNREFERENCED_PARAMETER(constantBuffer);};
21
22     void SetZoomLevel(const float level);
23     float GetZoomLevel();
24
25     void SetAttachedToParent(const bool value);
26     bool getAttachedToParent();
27
28     XMFLOAT4X4 GetViewMatrix();
29     XMFLOAT4X4 GetProjectionMatrix();
30
31     Vector3 GetPosition();
32
33     void UpdateView();
34     void UpdateProj();
35 private:
36
37     bool attachedToParent;
38
39     XMFLOAT4X4 view;
40     XMFLOAT4X4 proj;
41     float zoom = 1;
42
43     Vector3 prevPos;
44 };
45
```

11.16 CParticleEmitter.h File Reference

Allows a entity to emit particles.

```
#include "Cerberus/Core/CComponent.h"
#include "Cerberus/Core/CEntity.h"
#include "Cerberus/Core/Entities/CParticle.h"
#include "Cerberus/Core/Utility/Math/Math.h"
#include <vector>
```

Classes

- class [CParticleEmitter](#)

11.16.1 Detailed Description

Allows a entity to emit particles.

Author

Luke Whiting

Date

May 2022

11.17 CParticleEmitter.h

[Go to the documentation of this file.](#)

```
1  /*****
8  #pragma once
9  #include "Cerberus/Core/CComponent.h"
10 #include "Cerberus/Core/CEntity.h"
11 #include "Cerberus/Core/Entities/CParticle.h"
12 #include "Cerberus/Core/Utility/Math/Math.h"
13 #include <vector>
14
15 class CParticleEmitter : public CComponent
16 {
17 public:
18     CParticleEmitter();
19     ~CParticleEmitter();
20
21     void SetTexture(const std::string& path);
22     void SetSize(const int size);
23
24     void UseRandomDirection(bool toggle, const Vector3 min, const Vector3 max);
25     void UseRandomVelocity(bool toggle, const float min, const float max);
26     void UseRandomLifetime(bool toggle, const float min, const float max);
27
28     void SetDirection(const Vector3 dir);
29     Vector3 GetDirection();
30
31     void SetVelocity(const float velo);
32     float GetVelocity();
33
34     void SetLifetime(const float life);
35     float GetLifetime();
36
37     void Start();
```

```

38     void Stop();
39
40     //Updated automatically every single frame
41     virtual void Update(float deltaTime);
42     virtual void Draw(struct ID3D11DeviceContext* context, const XMFLLOAT4X4& parentMat, ConstantBuffer
43         cb, ID3D11Buffer* constantBuffer);
44 private:
45
46     std::vector<CParticle*> particles;
47     bool emit;
48
49
50     // Set Overall Variables.
51     Vector3 overallDirection;
52     float overallVelocity;
53     float overallLifetime;
54     std::string overallTexturePath;
55
56     // Random Variables
57     bool useRandDir;
58     bool useRandVelo;
59     bool useRandLife;
60
61     Vector3 randDirMin;
62     Vector3 randDirMax;
63
64     float randVeloMin;
65     float randVeloMax;
66
67     float randLifeMin;
68     float randLifeMax;
69 };
70

```

11.18 CRigidBodyComponent.cpp File Reference

Adds basic rigid body physics to a entity.

```

#include "CRigidBodyComponent.h"
#include "Cerberus/Core/CEntity.h"

```

11.18.1 Detailed Description

Adds basic rigid body physics to a entity.

Author

Luke Whiting

Date

Jan 2022

11.19 CRigidBodyComponent.h

```

1 #pragma once
2 #include "Cerberus/Core/CComponent.h"
3 class CRigidBodyComponent : public CComponent
4 {
5 public:
6     CRigidBodyComponent();
7     virtual ~CRigidBodyComponent();
8
9     virtual void Update(float deltaTime);
10    virtual void Draw(struct ID3D11DeviceContext* context, const XMFLLOAT4X4& parentMat, ConstantBuffer
        cb, ID3D11Buffer* constantBuffer)
11    {
12        UNREFERENCED_PARAMETER(context);
13        UNREFERENCED_PARAMETER(parentMat);
14        UNREFERENCED_PARAMETER(cb);
15        UNREFERENCED_PARAMETER(constantBuffer);
16    };
17
18    void SetVelocity(const Vector3& velo);
19    Vector3& GetVelocity();
20
21    void SetAcceleration(const Vector3& accel);
22    Vector3& GetAcceleration();
23
24 private:
25     float damping;
26     Vector3 acceleration;
27     Vector3 velocity;
28 };
29

```

11.20 CSpriteComponent.h File Reference

A component for loading and displaying a 2D texture in world space as part of [CEntity](#).

```

#include "Cerberus\Core\CComponent.h"
#include "Cerberus\Core\Structs\CMesh.h"
#include "Cerberus\Core\Structs\CTexture.h"
#include "Cerberus\Core\Structs\CMaterial.h"

```

Classes

- class [CSpriteComponent](#)

A component for loading and displaying a 2D texture in world space as part of [CEntity](#).

11.20.1 Detailed Description

A component for loading and displaying a 2D texture in world space as part of [CEntity](#).

Author

Arrien Bidmead

Date

January 2022

11.21 CSpriteComponent.h

[Go to the documentation of this file.](#)

```
1  /*****
9  #pragma once
10 #include "Cerberus\Core\CComponent.h"
11 #include "Cerberus\Core\Structs\CMesh.h"
12 #include "Cerberus\Core\Structs\CTexture.h"
13 #include "Cerberus\Core\Structs\CMaterial.h"
14
18 class CSpriteComponent : public CComponent
19 {
20     CMesh* mesh = nullptr;
21     CMaterial* material = nullptr;
22     CTexture* texture = nullptr;
23
24     XMUINT2 renderRect;
25     XMFLOAT2 textureOffset = { 0,0 };
26     XMUINT2 spriteSize;
27     XMFLOAT4 tint = { 0,0,0,0 };
28
29 public:
30
35     virtual void SetRenderRect(const XMUINT2& newSize);
36
42     void SetTextureOffset(const XMFLOAT2& newOffset);
43
48     virtual void SetSpriteSize(const XMUINT2& newSize) { spriteSize = newSize; };
49
53     void SetTint(const XMFLOAT4& newTint);
54
55     virtual void SetUseTranslucency(const bool& newTranslucency) override;
56
61     HRESULT LoadTexture(const std::string& filePath);
62
67     HRESULT LoadTextureWIC(const std::string& filePath);
68
69     const XMUINT2& GetRenderRect() const { return renderRect; };
70     const XMFLOAT2& GetTextureOffset() const { return textureOffset; };
71     const XMUINT2& GetSpriteSize() const { return spriteSize; };
72     const XMFLOAT4& GetTint() const { return tint; };
73     const XMUINT2& GetTextureSize() const { if (texture != nullptr) return texture->textureSize; else
74     return { 0,0 }; };
75     virtual XMFLOAT4X4 GetTransform() override;
76
77     CSpriteComponent();
78     virtual void Update(float deltaTime) override;
79     virtual void Draw(ID3D11DeviceContext* context, const XMFLOAT4X4& parentMat, ConstantBuffer cb,
80     ID3D11Buffer* constantBuffer) override;
81     virtual ~CSpriteComponent();
82 };
```

11.22 CTextRenderComponent.h File Reference

A component for rendering text to the screen from a sprite-sheet.

```
#include "Cerberus\Core\Components\CSpriteComponent.h"
```

Classes

- class [CTextRenderComponent](#)

A component for rendering text to the screen from a sprite-sheet.

Enumerations

- enum class [TextJustification](#) { **Right** , **Center** , **Left** }

An enum for how text will be justified relative to the component origin.

11.22.1 Detailed Description

A component for rendering text to the screen from a sprite-sheet.

Author

Arrien Bidmead

Date

January 2022

11.22.2 Enumeration Type Documentation

11.22.2.1 TextJustification

```
enum class TextJustification [strong]
```

An enum for how text will be justified relative to the component origin.

Like in MSWord where right justified text is default.

11.23 CTextRenderComponent.h

[Go to the documentation of this file.](#)

```
1 /*****/
9 #pragma once
10 #include "Cerberus\Core\Components\CSpriteComponent.h"
11
16 enum class TextJustification
17 {
18     Right, Center, Left
19 };
20
24 class CTextRenderComponent : public CComponent
25 {
26     std::string text = "";
27     std::string font = "Resources/Engine/font.png";
28     std::vector<CSpriteComponent*> sprites;
29     XMUINT2 characterSize = { 7,7 };
30     XMUINT2 characterDrawSize = { 14,14 };
31     unsigned short reserveSpriteCount = 16;
32     unsigned short usedSpriteCount = 0;
33     TextJustification justification = TextJustification::Center;
34     unsigned short spriteSheetColumns = 16;
35
36 public:
40     HRESULT SetFont(std::string filePath);
41
45     void SetText(std::string newText);
46
51     void SetReserveCount(unsigned short newReserveCount);
52
58     void SetJustification(TextJustification newJustification);
59
65     void SetCharacterSize(XMUINT2 newSize);
66
70     void SetCharacterDrawSize(XMUINT2 newSize);
71
77     void SetSpriteSheetColumnsCount(unsigned short newColumnsCount);
78
```

```

79     const std::string& GetText() const { return text; };
80     const unsigned short& GetReserveCount() const { return reserveSpriteCount; };
81     const XMUINT2& GetCharacterSize() const { return characterSize; };
82     const XMUINT2& GetCharacterDrawSize() const { return characterDrawSize; };
83     const unsigned short& SetSpriteSheetColumnsCount() const { return spriteSheetColumns; };
84
85     CTextRenderComponent();
86     virtual void Update(float deltaTime) override;
87     virtual void Draw(ID3D11DeviceContext* context, const XMFLLOAT4X4& parentMat, ConstantBuffer cb,
88                     ID3D11Buffer* constantBuffer) override;
89     virtual ~CTextRenderComponent();
90 };

```

11.24 Engine.h

```

1  #pragma once
2
3  #include <windows.h>
4  #include <windowsx.h>
5  #include <d3d11_1.h>
6  #include <d3dcompiler.h>
7  #include <directxmath.h>
8  #include <directxcOLORS.h>
9  #include <DirectXCollision.h>
10 #include <vector>
11 #include <iostream>
12
13 #include "Cerberus\Dependencies\Microsoft\DDSTextureLoader.h"
14
15 #pragma warning(push)
16 //Disabled Warnings that reside in external libraries.
17 #pragma warning(disable : 26812)
18 #include "Cerberus\Dependencies\Microsoft\WICTextureLoader.h"
19 #pragma warning(pop)
20
21
22 #include "Cerberus\Dependencies\IMGUI\imgui.h"
23 #include "Cerberus\Dependencies\IMGUI\imgui_impl_dx11.h"
24 #include "Cerberus\Dependencies\IMGUI\imgui_impl_win32.h"
25
26 #include "Cerberus/Core/Utility/DebugOutput/Debug.h"
27 #include "Cerberus/Core/Utility/InputManager/InputManager.h"
28 #include "Cerberus/Core/Utility/EntityManager.h"
29
30 #include "Cerberus/Core\Structs\structures.h"
31 #include "Cerberus\Resource.h"
32
33 #define PI 3.14159
34 #define DEG2RAD PI / 180
35 #define RAD2DEG 180 / PI
36
37 #define NAME_OF( v ) #v
38
39 class CEntity;
40 class CCameraComponent;
41
42 struct Engine
43 {
44     static bool Start(HINSTANCE hInstance, int nCmdShow, WNDPROC wndProc);
45
46     static void RenderUpdateLoop();
47
48     static LRESULT ReadMessage(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);
49
50     static void Stop();
51
52     static void SetRenderCamera(CCameraComponent* cam);
53
54     // Returns all entities of provided type that exist in the engine.
55     template<class T>
56     static std::vector<T*> GetEntityOfType()
57     {
58         std::vector<T*> outputVector;
59
60         for (size_t i = 0; i < EntityManager::GetEntitiesVector()->size(); i++)
61         {
62             T* e = dynamic_cast<T*>(EntityManager::GetEntitiesVector()->at(i));
63             if (e != nullptr)
64             {
65                 outputVector.push_back(e);
66             }
67         }
68     }
69 }

```

```

68
69         return outputVector;
70     };
71
72     static void DestroyEntity(CEntity* targetEntity);
73
74     template<class T>
75     // Creates a entity, adds it to drawables and returns it back.
76     static T* CreateEntity()
77     {
78         CEntity* temp = new T();
79         EntityManager::AddEntity(temp);
80         return (T*)temp;
81     };
82
83     // Window and Instance.
84     static HINSTANCE instanceHandle;
85     static HWND windowHandle;
86     static unsigned int windowWidth;
87     static unsigned int windowHeight;
88
89     // Direct3D.
90     static D3D_DRIVER_TYPE driverType;
91     static D3D_FEATURE_LEVEL featureLevel;
92     static ID3D11Device* device;
93     static ID3D11DeviceContext* deviceContext;
94
95     static XMMATRIX projMatrixUI;
96
97     static bool paused;
98 };

```

11.25 CParticle.cpp File Reference

A helper class for the ParticleEmitter, encapsulates a singlar particle that is emitted.

```
#include "CParticle.h"
```

11.25.1 Detailed Description

A helper class for the ParticleEmitter, encapsulates a singlar particle that is emitted.

Author

Luke Whiting

Date

May 2022

11.26 CParticle.h

```

1  /*****
2  #pragma once
3  #include "Cerberus/Core/CEntity.h"
4  #include "Cerberus/Core/Components/CSpriteComponent.h"
5  #include "Cerberus/Core/Utility/Vector3.h"
6
7  class CParticle : public CEntity
8  {
9  public:
10     CParticle();
11     ~CParticle();
12
13     virtual void Update(float deltaTime);

```

```

20     void Draw(ID3D11DeviceContext* context, const XMFLOAT4X4& parentMat, ConstantBuffer cb, ID3D11Buffer*
        constantBuffer);
21
22     void SetLifetime(const float life);
23     float GetLifetime();
24
25     void SetVelocity(const float velo);
26     float GetVelocity();
27
28     void SetDirection(const Vector3 dir);
29     Vector3 GetDirection();
30
31     CSpriteComponent* getSpriteComponent();
32
33 private:
34     CSpriteComponent* sprite;
35     Vector3 direction;
36     float lifetime;
37     float velocity;
38 };
39

```

11.27 CGridCursor.h

```

1 #pragma once
2 #include "Cerberus\Core\CEntity.h"
3
4 class CGridCursor :
5     public CEntity
6 {
7 public:
8     CGridCursor();
9
10     class CSpriteComponent* activeCellSprite = nullptr;
11
12
13
14     virtual void Update(float deltaTime) override;
15
16     void UpdateSize(int X, int Y);
17
18     Vector3 Offset;
19     Vector3 Offset_Start;
20     Vector3 Offset_End;
21
22     bool screenMoved;
23
24
25
26     bool cellInspectingEntity;
27
28
29
30     bool cellSelected;
31     Vector3 selectedCell_1;
32     bool wasMouseReleased;
33
34     class CCameraComponent* camera;
35
36 };
37

```

11.28 CTile.h

```

1 #pragma once
2 #include "Cerberus\Core\Utility\Vector3.h"
3 #include "Cerberus\Core\CEntity.h"
4 #include "Cerberus\WorldConstants.h"
5
6 enum class TileType
7 {
8     Floor,
9     Wall,
10    Door
11 };
12
13
14 class CTile : public CEntity
15 {

```

```

16 public:
17     CTile();
18     CTile(int TileID, Vector3 Position);
19     class CSpriteComponent* sprite = nullptr;
20     class CSpriteComponent* debugSprite = nullptr;
21
22
23
24     virtual void Update(float deltaTime) override;
25     virtual ~CTile();
26
27
28
29
30     void ChangeTileID(CellID TileID);
31     void ChangeTileID(int ID)
32     {
33         ChangeTileID(static_cast<CellID>(ID));
34     }
35     int GetTileID() { return tileId; }
36
37
38
39     std::vector<int> GetConnectedTiles() { return connectedTiles; }
40
41
42     void AddConnectedTile(int Tile) { connectedTiles.push_back(Tile); }
43
44
45     void SetNavID(int ID) { navId = ID; }
46
47     int GetNavID() { return navId; }
48
49     bool IsWalkable() { return isWalkable; }
50
51
52
53     void SetDebugMode(bool newState);
54
55     void UpdateDebugRender();
56
57 protected:
58
59     //Returns the tile's type, whether it be a walkable floor, a wall or a door.
60     TileType GetTileType() { return tileStatus; }
61
62
63 private:
64
65     bool debugMode = false;
66
67     bool isWalkable = false;
68
69     void SetRenderData(int X, int Y);
70
71
72
73
74     TileType tileStatus = TileType::Floor;
75
76     int tileId = -1;
77
78     int navId = -1;
79
80     std::vector<int> connectedTiles;
81
82
83
84
85
86
87
88 };
89

```

11.29 CWorld.h

```

1 #pragma once
2
3 #include <string>
4 #include <vector>
5 #include "CTile.h"
6

```

```

7 #include "Cerberus\WorldConstants.h"
8
9
10 #include "Necrodoggiecon/Game/AI/AlarmEnemy.h"
11 #include "Necrodoggiecon/Game/AI/GruntEnemy.h"
12 #include "Necrodoggiecon/Game/AI/DogEnemy.h"
13 #include "Cerberus\Dependencies\NlohmannJson\json.hpp"
14
15 using json = nlohmann::json;
16
17 class CWorld
18 {
19
20 public:
21     CWorld();
22     CWorld(int Slot);
23
24     int GetMapSlot() { return mapSlot; }
25
26
27
28     virtual void LoadWorld(int Slot);
29
30     //Extendable function, primarily used to setup unique level specific requirements, one of these
    things would be the editor peripheral
31     virtual void SetupWorld();
32
33     virtual void UnloadWorld();
34
35
36
37     virtual void ReloadWorld();
38
39     virtual void DestroyWorld();
40
41
42
43
44     //A List of all tiles in the scene
45     //std::vector<Tile*> tileList;
46
47
48
49     // TODO- Add collision collector
50     CTile* GetTileByID(int ID) { return tileContainer[ID]; }
51
52     std::vector<CTile*> GetAllWalkableTiles();
53
54     std::vector<CTile*> GetAllObstacleTiles();
55
56     void BuildNavigationGrid();
57
58     void AddEntityToList(class CEntity* NewEntity) { EntityList.push_back(NewEntity); }
59
60 protected:
61
62
63
64     virtual void LoadEntities(int Slot);
65
66
67
68
69
70
71
72
73 protected:
74
75
76
77     int mapSize =
78         mapScale * mapScale;
79
80
81
82     //std::map<Vector3, CTile*> tileContainer;
83
84     CTile* tileContainer[mapScale * mapScale];
85
86
87     //Function that loads entities based on slot, You can change the entities in each slot inside the cpp
88     //static void LoadEntity(int Slot, Vector3 Position);
89
90     //This function should only be used when Loading / Reloading the scene.
91
92

```

```

93
94     //This is a list of entities loaded in with the level data. This should not be touched outside of
95     Loading / Reloading
96     //std::vector<CT_EntityData> storedEntities;
97
98     //List of entities spawned in by this class, used for deconstruction.
99     //static std::vector<class CEntity*> entityList;
100
101 protected:
102
103     Vector3 IndexToGrid(int ID);
104     int GridToIndex(Vector2 Position);
105
106
107
108     //The slot that the current map is tied to.
109     int mapSlot;
110
111     std::vector<CEntity*> EntityList;
112
113
114     Vector2 StartPos;
115
116
117
118 };
119
120
121
122

```

11.30 CWorld_Edit.h

```

1 #pragma once
2 #include "CWorld.h"
3 #include "Cerberus\Tools\CT_EditorEntity.h"
4
5
6 struct CellData
7 {
8     int id;
9     CellType type;
10 };
11
12 enum class EditOperationMode
13 {
14     None, Additive, Subtractive, Additive_Single, Subtractive_Single, Move_Entity, EnemyEntity, Waypoints,
15     WeaponHolder
16 };
17
18 struct PropData
19 {
20     std::string propName;
21     Vector2 collisionData;
22     Vector2 atlasSize;
23 };
24
25 class CWorld_Editable : public CWorld
26 {
27
28 public:
29
30
31
32     EditOperationMode GetOperationMode() { return operationType; }
33
34     //Set the current operation mode
35     void SetOperationMode(EditOperationMode mode);
36     void SetEntityID(int ID) { selectedEntityID = ID; }
37
38     //Adds a cell to the Queue, once the queue is full (2 Cells) the grid will perform a edit operation;
39     void QueueCell(Vector2 Cell);
40
41     //Sets the lock-State to the input parameter
42     void ToggleCellQueueLock(bool setLock) { isQueueLocked = setLock; }
43
44     //Clears the Cell edit queue
45     void ClearQueue();
46
47     void PerformOperation(Vector2 A, Vector2 B);
48

```



```

49 //Public wrapper for clear space, clears the queue.
50 void PerformOperation_ClearSpace();
51
52 //Loads the world and initialises TileData
53 virtual void LoadWorld(int Slot) override;
54 virtual void UnloadWorld() override;
55 virtual void SetupWorld();
56
57 //Save the current tile data to a file
58 void SaveWorld(int Slot);
59 //Run edit operations currently inside of the function. Automatically save afterwards.
60 void EditWorld(int Slot);
61
62 //Initialises the tileset to empty
63 void NewWorld(int Slot);
64
65
66 void ToggleDebugMode(bool isDebug);
67
68 void UpdateEditorViewport();
69
70
71
72
73 EditorEntityType GetInspectedItemType();
74 CT_EditorEntity* GetInspectedItem_Standard() { return inspectedEntity; }
75 class CT_EditorEntity_Enemy* GetInspectedItem_Enemy() { return
static_cast<CT_EditorEntity_Enemy*>(inspectedEntity); }
76 CT_EditorEntity_Waypoint* GetInspectedItem_Waypoint() { return
static_cast<CT_EditorEntity_Waypoint*>(inspectedEntity); }
77 CT_EditorEntity_WeaponHolder* GetInspectedItem_WeaponHolder() { return
static_cast<CT_EditorEntity_WeaponHolder*>(inspectedEntity); }
78
79 void ShouldInspectEntity(Vector2 MousePos);
80
81 void MoveSelectedEntity(Vector3 Position);
82
83 void RemoveSelectedEntity();
84 protected:
85
86
87
88 //Wrapper function for BoxOperation, Sets space to be unwalkable
89 void AdditiveBox(Vector2 A, Vector2 B);
90
91 //Wrapper function for BoxOperation, Sets space to be walkable
92 void SubtractiveBox(Vector2 A, Vector2 B);
93
94 //Wrapper function for BoxOperation, Sets space to be unwalkable
95 void AdditiveBox_Scale(Vector2 A, Vector2 B);
96
97 //Wrapper function for BoxOperation, Sets space to be walkable
98 void SubtractiveBox_Scale(Vector2 A, Vector2 B);
99
100 //Clears the grid and sets all to empty
101 void ClearSpace();
102
103 void Additive_Cell(Vector2 A);
104
105 void Subtractive_Cell(Vector2 A);
106
107 //Add Enemy enetity to the map
108 void AddEditorEntity_EnemyCharacter(Vector2 Position, int Slot);
109
110 void AddEditorEntity_Decoration(Vector2 Position, int Slot);
111
112 void AddEditorEntity_Waypoint(Vector2 Position);
113
114 void AddEditorEntity_Prop(int Slot);
115
116 void AddEditorEntity_WeaponHolder(Vector2 Position);
117
118
119 void GeneratePropList();
120
121 private:
122
123 //Performs an operation on the grid, drawing a retangular shape based on the two provided
coordinates.
124 void BoxOperation(Vector2 A, Vector2 B, int TileID);
125
126 //Generates the grid based on the current tile data state.
127 void GenerateTileMap();
128
129 //Sets any corner that qualifies as an edge to an Edge
130 bool SetCorner(Vector2 Position);
131

```

```

132
133
134
135
136
137
138     CellData tileData[mapScale * mapScale];
139
140     //CellType CellList[mapScale * mapScale];
141
142
143     //Is the selected tile adjacent to a walkable tile
144     bool IsFloorAdjacent(Vector2 Position);
145
146
147     //Is the Tile at provided position equal to the provided Type
148     bool IsTile(Vector2 Position, CellType Type)
149     {
150         return tileData[GridToIndex(Position)].type == Type;
151     }
152
153     // the Tile at the provided position the equivalent to wall. (Edge/InnerCorner/OuterCorner)
154     bool IsEdge(Vector2 Pos)
155     {
156         return (tileData[GridToIndex(Pos)].type == CellType::Edge || tileData[GridToIndex(Pos)].type ==
CellType::OuterCorner || tileData[GridToIndex(Pos)].type == CellType::InnerCorner);
157     }
158
159     //Returns total amount of the given type of tile adjacent to the given tile.
160     int GetTotalAdjacentsOfType(Vector2 Pos, CellType AdjacentType);
161
162
163     //Gets the direction of adjacent tiles that match the given type.
164     // 2 = Both sides
165     // 1 = positive direction
166     // -1 = negative direction
167     Vector2 FindAdjacents(Vector2 Pos, CellType ID);
168
169     //Same as standard version but only returns the results for adjacent walls
170     Vector2 FindAdjacentEdges(Vector2 Pos);
171
172     //Gets adjacent diagonal tiles
173     //Only returns the first result
174     Vector2 FindFloorAdjacentDiagonal(Vector2 Position);
175
176     bool IsTileOccupied(Vector2 Pos);
177
178
179
180 private:
181
182
183     //Current edit mode
184     EditOperationMode operationType;
185
186     //Cached position for the current edit operation
187     Vector2 editOrigin;
188
189
190     //Whether or not an operation is taking place
191     bool selectedCell;
192
193     //Whether or not any edit operations can be performed
194     bool isQueueLocked;
195
196     //main editor viewport
197     class CT_EditorMain* editorViewport;
198
199     //The ID of the selected entity brush, used to place entities from the content panel
200     int selectedEntityID;
201
202     //The entity currently being inspected
203     CT_EditorEntity* inspectedEntity;
204
205     //Total number of enemy entnties used for saving
206     int totalEnemyEntities;
207     //Total number of enemy entities used for saving
208     int totalPropEntities;
209
210     class CT_EditorEntity* playerStartEntity;
211
212     //Full list of all editor entities
213     std::vector<class CT_EditorEntity*> editorEntityList;
214
215 };
216

```

11.31 IInputable.h

```

1 #pragma once
2
3 class IInputable
4 {
5 public:
6     virtual void PressedHorizontal(int dir, float deltaTime) = 0;
7     virtual void PressedVertical (int dir, float deltaTime) = 0;
8     virtual void PressedInteract() = 0;
9     virtual void PressedDrop() = 0;
10    virtual void Attack() = 0;
11    virtual void PressedUse() = 0;
12 };

```

11.32 CCamera.h File Reference

Class for storing all camera information needed for rendering.

```

#include "Cerberus\Core\Engine.h"
#include "Cerberus/Core/CEntity.h"

```

Classes

- class [CCamera](#)

11.32.1 Detailed Description

Class for storing all camera information needed for rendering.

Author

Arrien Bidmead

Date

January 2022

11.33 CCamera.h

[Go to the documentation of this file.](#)

```

1  /*****
9  #pragma once
10 #include "Cerberus\Core\Engine.h"
11 #include "Cerberus/Core/CEntity.h"
12
13 class CCamera : public CEntity
14 {
15 public:
16     CCamera() {};
17     virtual void Update(float deltaTime) { UNREFERENCED_PARAMETER(deltaTime); };
18 };

```

11.34 CMaterial.h File Reference

Holds the directx stuff for uploading sprite specific data to the shader.

```
#include "Cerberus\Core\Engine.h"
```

Classes

- struct [_Material](#)
- struct [MaterialPropertiesConstantBuffer](#)
- struct [CMaterial](#)

Holds the directx stuff for uploading sprite specific data to the shader.

11.34.1 Detailed Description

Holds the directx stuff for uploading sprite specific data to the shader.

Author

Arrien Bidmead

Date

January 2022

11.35 CMaterial.h

[Go to the documentation of this file.](#)

```
1  /*****/
9  #pragma once
10 #include "Cerberus\Core\Engine.h"
11
12 struct _Material
13 {
14     _Material()
15         : UseTexture(false)
16         , textureSize(0, 0)
17         , textureRect(0, 0)
18         , textureOffset(0, 0)
19         , tint(0, 0, 0, 0)
20         , padding2()
21         , padding1()
22         , translucent(false)
23     {}
24
25     int         UseTexture;
26     float       padding1[3];
27
28     XMUINT2     textureSize;
29     XMUINT2     textureRect;
30
31     XMFLOAT2     textureOffset;
32     int         translucent;
33     float       padding2;
34
35     XMFLOAT4     tint;
36 };
37
38 struct MaterialPropertiesConstantBuffer
39 {
40     _Material    Material;
```

```

41 };
42
43 struct CMaterial
44 {
45     MaterialPropertiesConstantBuffer material;
46     ID3D11Buffer* materialConstantBuffer = nullptr;
47     bool loaded = false;
48
49     CMaterial();
50     HRESULT CreateMaterial(XMUINT2 texSize);
51     void UpdateMaterial();
52     ~CMaterial();
53 };
54
55

```

11.36 CMesh.h File Reference

Holds all information about a mesh for use by [CSpriteComponent](#).

```
#include "Cerberus\Core\Engine.h"
```

Classes

- struct [SimpleVertex](#)
- struct [CMesh](#)

Holds all information about a mesh for use by [CSpriteComponent](#).

11.36.1 Detailed Description

Holds all information about a mesh for use by [CSpriteComponent](#).

Author

Arrien Bidmead

Date

January 2022

11.37 CMesh.h

[Go to the documentation of this file.](#)

```

1  /*****
9  #pragma once
10 #include "Cerberus\Core\Engine.h"
11
12 struct SimpleVertex
13 {
14     XMFLOAT3 Pos;
15     XMFLOAT2 TexCoord;
16 };
17
22 struct CMesh
23 {
24     ID3D11Buffer* vertexBuffer;
25     ID3D11Buffer* indexBuffer;
26
27     bool loaded = false;
28
29     CMesh();
30     HRESULT LoadMesh();
31     ~CMesh();
32 };
33

```

11.38 CTexture.h File Reference

Holds all information about a texture for use by [CSpriteComponent](#).

```
#include "Cerberus\Core\Engine.h"
```

Classes

- struct [CTexture](#)

Holds all information about a texture for use by [CSpriteComponent](#).

11.38.1 Detailed Description

Holds all information about a texture for use by [CSpriteComponent](#).

Author

Arrien Bidmead

Date

January 2022

11.39 CTexture.h

[Go to the documentation of this file.](#)

```
1  /*****
9  #pragma once
10 #include "Cerberus\Core\Engine.h"
11
16 struct CTexture
17 {
18     XMUINT2 textureSize = {0,0};
19
20     ID3D11ShaderResourceView* textureResourceView;
21     ID3D11SamplerState* samplerLinear;
22     bool loaded = false;
23
24     CTexture();
25     HRESULT LoadTextureDDS(std::string filePath);
26     HRESULT LoadTextureWIC(std::string filename);
27     ~CTexture();
28 };
```

11.40 structures.h

```
1 #pragma once
2
3 using namespace DirectX;
4
5 //-----
6 // Structures
7 //-----
8
9 struct ConstantBuffer
10 {
11     XMMATRIX mWorld;
12     XMMATRIX mView;
13     XMMATRIX mProjection;
14     XMFLOAT4 vOutputColor;
15 };
```

11.41 CWidget.cpp File Reference

Base class for all UI widgets.

```
#include "Cerberus/Core/UI/CWidget.h"
```

11.41.1 Detailed Description

Base class for all UI widgets.

Handles parenting operations

Author

Samuel Elliot Jackson

Date

May 2022

11.42 CWidget.h

```
1 #pragma once
2 #include "Cerberus/Core/CEntity.h"
3 class CWidget :
4     public CEntity
5 {
6 public:
7     CWidget();
8
9     CWidget* GetParent() { return parentWidget; }
10
11     const std::vector<CWidget*> GetChildren() { return childWidgets; }
12
13
14
15     virtual void SetWidgetTransform(Vector2 Position, Vector2 Anchor, int ZOrder);
16
17     virtual void SetVisibility(bool isVisible);
18
19     void AddChild(CWidget* NewChild);
20
21     void RemoveAllChildren();
22
23
24     void UpdateWidgetOrigin(Vector3 Pos);
25
26 private:
27     CWidget* parentWidget;
28
29     std::vector<CWidget*> childWidgets;
30
31 protected:
32     bool WidgetIsVisible = true;
33
34 };
35
```

11.43 CWidget_Button.cpp File Reference

Button Widget class, provides all functionality for buttons and allows to functions to be bound to button events.

```
#include "Cerberus/Core/UI/CWidget_Button.h"
#include "Cerberus/Core/Components/CSpriteComponent.h"
#include "Cerberus/Core/Components/CTextRenderComponent.h"
#include "Cerberus/Core/UI/CWidget_Canvas.h"
```

11.43.1 Detailed Description

Button Widget class, provides all functionality for buttons and allows to functions to be bound to button events.

Author

Samuel Elliot Jackson

Date

May 2022

11.44 CWidget_Button.h

```

1  /*****
2  #pragma once
3  #include "Cerberus/Core/UI/CWidget.h"
4  #include <functional>
5
6  class CWidget_Button :
7      public CWidget
8  {
9  public:
10
11     CWidget_Button();
12
13     void SetText(std::string TextBody);
14     void SetButtonSize(Vector2 Size);
15     void SetTexture(std::string filePath);
16
17     virtual void SetWidgetTransform(Vector2 Position, Vector2 Anchor, int ZOrder);
18
19     virtual void Update(float deltaTime) override;
20
21     virtual void OnButtonPressed();
22     virtual void OnButtonReleased();
23
24     virtual void OnButtonHoverStart();
25     virtual void OnButtonHoverEnd();
26
27     virtual void SetVisibility(bool isVisible);
28
29     void IsButtonFocused(Vector2 mPos);
30     void ButtonPressed(bool buttonPressed);
31
32     void Bind_OnButtonPressed(std::function<void()> functionToBind) { ButtonPressedBind = functionToBind; }
33     void Bind_OnButtonReleased(std::function<void()> functionToBind) { ButtonReleasedBind = functionToBind; }
34     void Bind_HoverStart(std::function<void()> functionToBind) { HoverStartBind = functionToBind; }
35     void Bind_HoverEnd(std::function<void()> functionToBind) { ButtonReleasedBind = functionToBind; }
36
37     class CSpriteComponent* GetSprite() { return sprite; }
38     class CTextRenderComponent* GetText() { return textRenderer; }
39
40     bool ButtonHasFocus() { return hasFocus; }
41
42 private:
43     Vector2 spriteSize;
44
45     std::function<void()> HoverStartBind;
46     std::function<void()> HoverEndBind;
47
48     std::function<void()> ButtonPressedBind;
49     std::function<void()> ButtonReleasedBind;
50
51     int buttonSlot;
52     bool hasFocus;
53
54     bool ButtonHeld = false;

```



```
113
114     class CWidget_Canvas* owningCanvas;
115
116     class CSpriteComponent* sprite = nullptr;
117     class CTextRenderComponent* textRenderer = nullptr;
118
119
120
121
122
123
124
125 };
126
```

11.45 CWidget_Canvas.h File Reference

Main container for all widget classes.

```
#include "Cerberus/Core/UI/CWidget.h"
#include <functional>
#include "Cerberus/Core/Components/CSpriteComponent.h"
#include "Cerberus/Core/Components/CTextRenderComponent.h"
```

Classes

- class [CWidget_Canvas](#)

11.45.1 Detailed Description

Main container for all widget classes.

If a widget is being used it should be instantiated through this object first. This enables easy access and tidy management.

Author

Samuel Elliot Jackson

Date

May 2022

11.48 CWidget_Image.h

[Go to the documentation of this file.](#)

```
1 /*****
2  *
3  *
4  *
5  *
6  *
7  *
8  *
9  *
10 *
11 *
12 *
13 *
14 *
15 *
16 *
17 *
18 *
19 *
20 *
21 *
22 *
23 *
24 *
25 *
26 *
27 *
28 *
29 *
30 *
31 *
32 *
33 *
34 *
35 *
36 *
37 *
38 *
39 *
40 *
41 *
42 *
43 *
44 *
45 *
46 *
47 */
8 #pragma once
9 #include "Cerberus/Core/UI/CWidget.h"
10 class CWidget_Image :
11     public CWidget
12 {
13
14
15 public:
16     CWidget_Image();
17
18
19
20
21     virtual void Update(float deltaTime) override;
22
23     virtual void SetWidgetTransform(Vector2 Position, Vector2 Anchor, int ZOrder);
24
25     class CSpriteComponent* GetSprite() { return sprite; }
26     class CTextRenderComponent* GetText() { return textRenderer; }
27
28     void SetSpriteData(Vector2 SpriteSize, std::string filePath);
29
30     virtual void SetVisibility(bool isVisible);
31
32 protected:
33     class CSpriteComponent* sprite = nullptr;
34     class CTextRenderComponent* textRenderer = nullptr;
35
36 };
47
```

11.49 CWidget_Text.h

```
1 #pragma once
2 #include "Cerberus/Core/UI/CWidget.h"
3 class CWidget_Text : public CWidget
4 {
5 public:
6     CWidget_Text();
7
8     virtual void Update(float deltaTime) override;
9
10    virtual void SetWidgetTransform(Vector2 Position, Vector2 Anchor, int ZOrder);
11
12    virtual void SetVisibility(bool isVisible);
13
14    class CTextRenderComponent* GetText() { return textRenderer; }
15
16 protected:
17    class CTextRenderComponent* textRenderer = nullptr;
18
19 };
20
21
```

11.50 AssetManager.h File Reference

A asset manager that holds assets to be retrieved.

```
#include "Cerberus\Core\Structs\CMesh.h"
#include "Cerberus\Core\Structs\CTexture.h"
#include "Cerberus/Core/Utility/Audio/CAudio.h"
#include <string>
#include <sstream>
#include <map>
```

Classes

- class [AssetManager](#)

11.50.1 Detailed Description

A asset manager that holds assets to be retrieved.

This avoids the overhead of making duplicate assets across the program.

Author

Luke Whiting.

Date

May 2022

11.51 AssetManager.h

[Go to the documentation of this file.](#)

```
1  /*****/
9  #pragma once
10 #include "Cerberus/Core/Structs/CMesh.h"
11 #include "Cerberus/Core/Structs/CTexture.h"
12 #include "Cerberus/Core/Utility/Audio/CAudio.h"
13 #include <string>
14 #include <sstream>
15 #include <map>
16
17 class AssetManager
18 {
19 public:
20     static CMesh* AddMesh(std::string meshID, CMesh* mesh);
21     static CMesh* GetMesh(std::string meshID);
22     static CMesh* GetDefaultMesh();
23     static CTexture* GetTexture(std::string texturePath);
24     static CTexture* GetTextureWIC(std::string texturePath);
25     static CAudio* AddAudio(std::string audioPath, CAudio* audio);
26     static CAudio* GetAudio(std::string audioPath);
27     static void RemoveAudio(std::string audioPath);
28
29     static void Destroy();
30
31 private:
32     static std::map<std::string, CMesh*> meshes;
33     static std::map<std::string, CTexture*> textures;
34     static std::map<std::string, CAudio*> audios;
35 };
36
```

11.52 AudioController.h File Reference

Internal Audio Controller for the engine.

```
#include "Cerberus/Dependencies/FMOD/api/core/inc/fmod.hpp"
#include "Cerberus/Dependencies/FMOD/api/core/inc/fmod_errors.h"
#include "Cerberus/Core/CEntity.h"
#include "Cerberus/Core/Utility/DebugOutput/Debug.h"
#include "Cerberus/Core/Utility/AssetManager/AssetManager.h"
#include "Cerberus/Core/Utility/Audio/CEmitter.h"
#include "Cerberus/Core/Utility/Vector3.h"
#include "Cerberus/Core/Utility/CTransform.h"
```

Classes

- class [AudioController](#)

11.52.1 Detailed Description

Internal Audio Controller for the engine.

Author

Luke Whiting

Date

Jan 2022

11.53 AudioController.h

[Go to the documentation of this file.](#)

```

1  /*****
2  #pragma once
3  */
4  #pragma warning(push)
5  //Disabled Warnings that reside in external libraries.
6  #pragma warning(disable : 4505)
7  #pragma warning(disable : 26812)
8  #include "Cerberus/Dependencies/FMOD/api/core/inc/fmod.hpp"
9  #include "Cerberus/Dependencies/FMOD/api/core/inc/fmod_errors.h"
10 #pragma warning(pop)
11
12 #include "Cerberus/Core/CEntity.h"
13 #include "Cerberus/Core/Utility/DebugOutput/Debug.h"
14 #include "Cerberus/Core/Utility/AssetManager/AssetManager.h"
15 #include "Cerberus/Core/Utility/Audio/CEmitter.h"
16 #include "Cerberus/Core/Utility/Vector3.h"
17 #include "Cerberus/Core/Utility/CTransform.h"
18
19 class AudioController
20 {
21 public:
22     static void Initialize();
23     static void Shutdown();
24
25     static CAudio* LoadAudio(const std::string& path);
26     static bool PlayAudio(const std::string& path);
27     static bool PlayAudio(const std::string& path, bool loop);
28     static bool StopAudio(const std::string& path);
29     static bool DestroyAudio(const std::string& path);
30
31     static void Update(float deltaTime);
32
33     static std::vector<CEmitter*> GetAllEmittersWithinRange(Vector3 position, bool checkIfPlaying);
34     static bool AddEmitter(CEmitter* emitter);
35     static bool RemoveEmitter(CEmitter* emitter);
36
37     static void SetMaxVolumeForEmitterType(const float volume, EMITTERTYPE type);
38
39     static bool AddListener(CTransform* listenerPos);
40
41 private:
42     static FMOD::System* FMODSystem;
43     static std::vector<CEmitter*> emitters;
44     static CTransform* listenerTransform;
45 };

```

11.54 CAudio.h File Reference

Helper class that encapsulates audio parameters for the audio system.

```
#include "Cerberus/Dependencies/FMOD/api/core/inc/fmod.hpp"
```

Classes

- class [CAudio](#)

11.54.1 Detailed Description

Helper class that encapsulates audio parameters for the audio system.

Used to de-couple FMOD from the audio system.

Author

Luke Whiting

Date

Jan 2022

11.55 CAudio.h

[Go to the documentation of this file.](#)

```
1  /*****/
8  #pragma once
9  #include "Cerberus/Dependencies/FMOD/api/core/inc/fmod.hpp"
10 class CAudio
11 {
12 public:
13     CAudio(std::string path, FMOD::Sound* sound, FMOD::ChannelGroup* group) : sound(sound), group(group),
        channel(nullptr), maxVolume(100) {};
14     CAudio(std::string path, FMOD::Sound* sound, FMOD::ChannelGroup* group, FMOD::Channel* channel) :
        path(path), sound(sound), group(group), channel(channel), maxVolume(100) {};
15     std::string path;
16     FMOD::Sound* sound;
17     FMOD::ChannelGroup* group;
18     FMOD::Channel* channel;
19     float maxVolume;
20 };
```

11.56 CEmitter.h File Reference

A helper class to help encapsulate emitters that can be used by the audio system.

```
#include "Cerberus/Core/Utility/Vector3.h"
#include "Cerberus/Core/Utility/Audio/CAudio.h"
```

Classes

- class [CEmitter](#)

Enumerations

- enum class **EMITTERTYPE** { SFX = 0 , AMBIENT , ALL }

11.56.1 Detailed Description

A helper class to help encapsulate emitters that can be used by the audio system.

Different from the audio emitter component.

Author

Luke Whiting

Date

Jan 2022

11.57 CEmitter.h

[Go to the documentation of this file.](#)

```
1 /*****
2 8 #pragma once
3 9 #include "Cerberus\Core\Utility\Vector3.h"
4 10 #include "Cerberus/Core/Utility/Audio/CAudio.h"
5 11
6 12 enum class EMITTERTYPE
7 13 {
8 14     SFX = 0,
9 15     AMBIENT,
10 16     ALL
11 17 };
12 18
13 19 class CEmitter
14 20 {
15 21 public:
16 22     Vector3 position;
17 23     float range = 1000;
18 24     CAudio* audio;
19 25     EMITTERTYPE type;
20 26 };
```

11.58 CameraManager.h File Reference

Manages the cameras in the engine.

```
#include <map>
#include <vector>
#include "Cerberus\Core\Components\CCameraComponent.h"
```

Classes

- class [CameraManager](#)

11.58.1 Detailed Description

Manages the cameras in the engine.

Author

Luke Whiting

Date

May 2022

11.59 CameraManager.h

[Go to the documentation of this file.](#)

```
1  /*****
9  #pragma once
10 #include <map>
11 #include <vector>
12 #include "Cerberus\Core\Components\CCameraComponent.h"
13 class CameraManager
14 {
15 public:
16
17     static void AddCamera(CCameraComponent* camera);
18     static void RemoveCamera(CCameraComponent* camera);
19     static CCameraComponent* GetRenderingCamera();
20     static void SetRenderingCamera(CCameraComponent* camera);
21     static std::vector<CCameraComponent*> GetAllCameras();
22
23 private:
24     static std::map<std::uintptr_t, CCameraComponent*> cameras;
25     static CCameraComponent* renderingCamera;
26 };
27
```

11.60 CollisionComponent.h

```
1 #pragma once
2 #include "Cerberus\Core\Utility\Vector3.h"
3 #include "Cerberus\Core\Utility\DebugOutput\Debug.h"
4 #include <thread>
5
6 enum class COLLISIONTYPE
7 {
8     BOUNDING_BOX,
9     BOUNDING_CIRCLE,
10    BOUNDING_NONE
11 };
12
13
14 class CEntity;
15
16 //A component for collisions
17 class CollisionComponent
18 {
19 public:
20     CollisionComponent(std::string setName, CEntity* parent);
21
22     ~CollisionComponent();
23
24     COLLISIONTYPE GetCollisionType();
```



```

25
26     float GetRadius();
27     void SetRadius(float setRadius);
28
29     void SetPosition(Vector3 setPosition);
30     Vector3 GetPosition();
31
32     std::string GetName() { return name; };
33
34     float GetWidth() { return width; };
35     float GetHeight() { return height; };
36
37     bool Intersects(CollisionComponent* circle, CollisionComponent* box);
38
39     void SetCollider(float setRadius); //Bounding circle initiation
40     void SetCollider(float setHeight, float setWidth); //Bounding Box initiation
41
42     bool IsColliding(CollisionComponent* collidingObject);
43     float DistanceBetweenPoints(Vector3& point1, Vector3& point2);
44
45     CEntity* GetParent();
46     void Resolve(CollisionComponent* other);
47
48     void SetTrigger(const bool value);
49     bool GetTrigger();
50
51 private:
52     float radius;
53     Vector3 position;
54     float height;
55     float width;
56     std::string name = "none";
57
58     bool trigger = false;
59
60     CEntity* parent = nullptr;
61
62     COLLISIONTYPE collisionType = COLLISIONTYPE::BOUNDING_NONE;
63 };
64

```

11.61 CTransform.h File Reference

A transform class that contains getters and setters.

```

#include "Cerberus\Core\Engine.h"
#include "Cerberus\Core\Utility\Vector3.h"

```

Classes

- class [CTransform](#)

A transform class that contains getters and setters.

11.61.1 Detailed Description

A transform class that contains getters and setters.

Author

Arrien Bidmead

Date

January 2022

11.62 CTransform.h

[Go to the documentation of this file.](#)

```

1  /*****
9  #pragma once
10 #include "Cerberus\Core\Engine.h"
11 #include "Cerberus\Core\Utility\Vector3.h"
12
16 class CTransform
17 {
18     Vector3 position = { 0,0,0 };
19     Vector3 scale = { 1,1,1 };
20     float rotation = 0;
21
22 protected:
23     bool updateTransform = true;    //use get transform instead of directly using this
24     XMFLOAT4X4 world = XMFLOAT4X4();
25
26 public:
27     void SetPosition(const float& x, const float& y, const float& z) { position = Vector3(x, y, z);
        updateTransform = true; }
28     void SetScale(const float& x, const float& y, const float& z) { scale = Vector3(x, y, z);
        updateTransform = true; }
29
30     void SetPosition(const Vector3& In) { position = In; updateTransform = true; }
31     void SetScale(const Vector3& In) { scale = In; updateTransform = true; }
32
33     void SetRotation(const float& Rot);
34
35     const Vector3& GetPosition() const { return position; }
36     const Vector3& GetScale() const { return scale; }
37     const float& GetRotation() const { return rotation; }
38
39     //Convert pos, scale and rot to a XMFloat4x4
40     virtual XMFLOAT4X4 GetTransform();
41 };

```

11.63 CUIManager.h

```

1 #include <map>
2 #include <string>
3 #include <vector>
4 #include "Cerberus/Core/Utility/Vector3.h"
5 #pragma once
6 class CUIManager
7 {
8     static std::map<std::string, class CWidget_Canvas*> activeCanvases;
9     static std::vector<std::string> idList;
10 public:
11
12
13
14     static class CWidget_Canvas* AddCanvas(class CWidget_Canvas* Canvas, std::string ID);
15
16     static void HideAllCanvases();
17
18     static class CWidget_Canvas* GetCanvas(std::string ID);
19
20     static void ClearAllCanvases();
21
22     static void UpdateUIOrigin(Vector3 Pos);
23
24
25
26
27 };
28

```

11.64 CWorldManager.h

```

1 #pragma once
2 #include "Cerberus/Core/Environment/CWorld_Edit.h"
3
4
5
6 class CWorldManager
7 {

```

```
8 public:
9     static void LoadWorld(int Slot, bool bEditorMode);
10    static void LoadWorld(CWorld* World);
11    static void LoadWorld(CWorld_Editable* World);
12
13    static void ReloadWorld();
14
15
16    static class CWorld* GetWorld() {
17        return gameWorld;
18    }
19
20    static class CWorld_Editable* GetEditorWorld() {
21        return editorWorld;
22    }
23
24
25
26 private:
27
28    static CWorld* gameWorld;
29    static CWorld_Editable* editorWorld;
30 };
31
```

11.65 Debug.h File Reference

Allows for debug logging to a in-game console using ImGui.

```
#include "Cerberus/Core/Utility/DebugOutput/DebugOutput.h"
#include <string>
#include <chrono>
#include <ctime>
#include <winerror.h>
#include <comdef.h>
```

Classes

- class [Debug](#)

11.65.1 Detailed Description

Allows for debug logging to a in-game console using ImGui.

Author

Luke Whiting

Date

Jan 2022

11.66 Debug.h

[Go to the documentation of this file.](#)

```

1  /*****
9  #pragma once
10 #include "Cerberus/Core/Utility/DebugOutput/DebugOutput.h"
11 #include <string>
12 #include <chrono>
13 #include <ctime>
14 #include <winerror.h>
15 #include <comdef.h>
16
17 class Debug
18 {
19
20 private:
21     static DebugOutput* output;
22     static int logSize;
23     static bool showDebug;
24     static bool allowLogging;
25     static void initOutput()
26     {
27         output = new DebugOutput();
28     }
29
30     // Helper function for getting the current system time into a std::string.
31     static std::string getCurrentTimeString()
32     {
33         // Get the current time
34         struct tm newtime;
35         time_t now = time(0);
36         localtime_s(&newtime, &now);
37
38         char buffer[8];
39         time(&now);
40
41         strftime(buffer, sizeof(buffer), "%H:%M", &newtime);
42         std::string timeString(buffer);
43
44         return "[" + timeString + "] ";
45     }
46
47     static void CheckLogSize()
48     {
49         if (output->getItems().size() > logSize)
50             output->ClearLog();
51     }
52
53 public:
54
55     //Disabled Warning for C4840, which is because the compiler doesnt like the fact im passing an
56     //varadic args to a varadic args.
57     #pragma warning(push)
58     #pragma warning(disable : 4840)
59
60     static void SetVisibility(bool value)
61     {
62         showDebug = value;
63     }
64
65     static bool GetVisibility()
66     {
67         return showDebug;
68     }
69
70     static void SetLogging(bool value)
71     {
72         allowLogging = value;
73     }
74
75     static bool GetLogging()
76     {
77         return allowLogging;
78     }
79
80     template<typename ... Args>
81     static void Log(const char* fmt, Args ... args) IM_FMTARGS(2)
82     {
83         if (!GetLogging())
84         {
85             return;
86         }
87
88         if (output == nullptr)

```

```

115         initOutput();
116
117         CheckLogSize();
118
119         std::string stringInput = std::string(fmt);
120
121         stringInput = getCurrentTimeString() + stringInput;
122
123         output->AddLog(stringInput.c_str(), args ...);
124     };
125
126     template<typename ... Args>
127     static void LogError(const char* fmt, Args ... args) IM_FMTARGS(2)
128     {
129         if (!GetLogging())
130         {
131             return;
132         }
133
134         if (output == nullptr)
135             initOutput();
136
137         CheckLogSize();
138
139         std::string stringInput = std::string(fmt);
140
141         stringInput = "[error] " + getCurrentTimeString() + stringInput;
142
143         output->AddLog(stringInput.c_str(), args ...);
144     };
145
146     template<typename ... Args>
147     static void LogHRESULT(HRESULT hr, const char* fmt, Args ... args) IM_FMTARGS(2)
148     {
149         if (!GetLogging())
150         {
151             return;
152         }
153
154         if (output == nullptr)
155             initOutput();
156
157         CheckLogSize();
158
159         std::string stringInput = "";
160
161         char* convOutput = nullptr;
162
163         if(FAILED(hr))
164         {
165             // Get the Error message out of the HRESULT.
166             _com_error err(hr);
167             LPCTSTR errMsg = err.ErrorMessage();
168             convOutput = new char[256];
169             size_t numConverted = 0;
170             size_t size = 256;
171
172             wcstombs_s(&numConverted, convOutput, size, errMsg, size-1);
173
174             std::string errorString = std::string(convOutput);
175
176             stringInput = "[HRESULT][error] " + getCurrentTimeString() + fmt + " " + errorString;
177         }else
178         {
179             stringInput = "[HRESULT]" + getCurrentTimeString() + fmt + " Completed Sucessfully.";
180         }
181         output->AddLog(stringInput.c_str(), args ...);
182
183         if (FAILED(hr))
184             delete[] convOutput;
185     }
186
187     #pragma warning(pop)
188
189     static DebugOutput* getOutput()
190     {
191         if (!output)
192             initOutput();
193
194         return output;
195     }
196
197     };
198
199     };
200
201     };
202
203     };
204
205     };
206
207     };
208
209     };
210
211     };
212
213     };
214
215     };
216 };

```

11.67 DebugOutput.h

```

1 #pragma once
2
3 #include "Cerberus\Dependencies\IMGUI\imgui.h"
4 #include "Cerberus\Dependencies\IMGUI\imgui_impl_dx11.h"
5 #include "Cerberus\Dependencies\IMGUI\imgui_impl_win32.h"
6 #include <corecrt_malloc.h>
7 #include <iostream>
8
9
10 /*
11
12     DEBUG CONSOLE TAKEN FROM IMGUI EXAMPLES. MODIFIED SLIGHTLY.
13
14 */
15
16 class DebugOutput
17 {
18     char                InputBuf[256];
19     ImVector<char*>      Items;
20     ImVector<const char*> Commands;
21     ImVector<char*>      History;
22     int                 HistoryPos;    // -1: new line, 0..History.Size-1 browsing history.
23     ImGuiTextFilter      Filter;
24     bool                AutoScroll;
25     bool                ScrollToBottom;
26     bool*               open;
27
28 public:
29
30     DebugOutput()
31     {
32         ClearLog();
33         memset(InputBuf, 0, sizeof(InputBuf));
34         HistoryPos = -1;
35
36         AutoScroll = true;
37         ScrollToBottom = false;
38         open = new bool(true);
39     }
40     ~DebugOutput()
41     {
42         ClearLog();
43         for (int i = 0; i < History.Size; i++)
44             free(History[i]);
45     }
46
47 private:
48
49     // Portable helpers
50     static int Stricmp(const char* s1, const char* s2) { int d; while ((d = toupper(*s2) - toupper(*s1)) == 0 && *s1) { s1++; s2++; } return d; }
51     static int Strnicmp(const char* s1, const char* s2, int n) { int d = 0; while (n > 0 && (d = toupper(*s2) - toupper(*s1)) == 0 && *s1) { s1++; s2++; n--; } return d; }
52     static char* Strdup(const char* s) { IM_ASSERT(s); size_t len = strlen(s) + 1; void* buf = malloc(len); IM_ASSERT(buf); return (char*)memcpy(buf, (const void*)s, len); }
53     static void Strtrim(char* s) { char* str_end = s + strlen(s); while (str_end > s && str_end[-1] == ' ') str_end--; *str_end = 0; }
54
55 public:
56
57     ImVector<char*> getItems() { return Items; }
58
59     void ClearLog()
60     {
61         for (int i = 0; i < Items.Size; i++)
62             free(Items[i]);
63         Items.clear();
64     }
65
66     // Use [error] to define errors.
67     void AddLog(const char* fmt, ...) IM_FMTARGS(2)
68     {
69         // FIXME-OPT
70         char buf[1024];
71         va_list args;
72         va_start(args, fmt);
73         vsnprintf(buf, IM_ARRAYSIZE(buf), fmt, args);
74         buf[IM_ARRAYSIZE(buf) - 1] = 0;
75         va_end(args);
76         Items.push_back(Strdup(buf));
77     }
78
79     void render()
80     {
81         if (*open)

```

```

82
83     {
84         ImGui::SetNextWindowSize(ImVec2(300, 120), ImGuiCond_FirstUseEver);
85         if (!ImGui::Begin("Debug Console", open))
86         {
87             ImGui::End();
88             return;
89         }
90
91         const float footer_height_to_reserve = ImGui::GetStyle().ItemSpacing.y +
ImGui::GetFrameHeightWithSpacing();
92         ImGui::BeginChild("ScrollingRegion", ImVec2(0, -footer_height_to_reserve), false,
ImGuiWindowFlags_HorizontalScrollbar);
93         if (ImGui::BeginPopupContextWindow())
94         {
95             if (ImGui::Selectable("Clear")) ClearLog();
96             ImGui::EndPopup();
97         }
98
99         ImGui::PushStyleVar(ImGuiStyleVar_ItemSpacing, ImVec2(4, 1)); // Tighten spacing
100        for (int i = 0; i < Items.Size; i++)
101        {
102            const char* item = Items[i];
103            if (!Filter.PassFilter(item))
104                continue;
105
106            // Normally you would store more information in your item than just a string.
107            // (e.g. make Items[] an array of structure, store color/type etc.)
108            ImVec4 color;
109            bool has_color = false;
110            if (strstr(item, "[error]")) { color = ImVec4(1.0f, 0.4f, 0.4f, 1.0f); has_color = true;
111        }
112            else if (strncmp(item, "# ", 2) == 0) { color = ImVec4(1.0f, 0.8f, 0.6f, 1.0f);
has_color = true; }
113            if (has_color)
114                ImGui::PushStyleColor(ImGuiCol_Text, color);
115            ImGui::TextUnformatted(item);
116            if (has_color)
117                ImGui::PopStyleColor();
118        }
119
120        if (ScrollToBottom || (AutoScroll && ImGui::GetScrollY() >= ImGui::GetScrollMaxY()))
121            ImGui::SetScrollHereY(1.0f);
122        ScrollToBottom = false;
123
124        ImGui::PopStyleVar();
125        ImGui::EndChild();
126
127        ImGui::Separator();
128
129        // Auto-focus on window apparition
130        ImGui::SetItemDefaultFocus();
131
132        ImGui::Text("Application average %.3f ms/frame (%.1f FPS)", 1000.0f /
ImGui::GetIO().Framerate, ImGui::GetIO().Framerate);
133
134    }
135 }
136 }
137 };
138

```

11.68 EntityManager.h File Reference

Static class for tracking entities and components while accommodating translucency.

```
#include <unordered_map>
```

Classes

- class [EntityManager](#)

Static class for tracking entities and components while accommodating translucency.

11.68.1 Detailed Description

Static class for tracking entities and components while accommodating translucency.

Author

Arrien Bidmead

Date

May 2022

11.69 EntityManager.h

[Go to the documentation of this file.](#)

```
1  /*****
9  #pragma once
10 #include <unordered_map>
11
15 class EntityManager
16 {
17     static std::vector<class CEntity*> entities;
18
19     static std::vector<class CComponent*> opaqueComps;
20     static std::vector<class CComponent*> translucentComps;
21
22 public:
26     static void AddEntity(class CEntity* entityToAdd);
27
32     static void RemoveEntity(const class CEntity* entityToRemove);
33
37     static void AddComponent(class CComponent* compToAdd);
38
43     static void RemoveComponent(const class CComponent* compToRemove);
44
49     static void SortTranslucentComponents();
50
51     static const std::vector<class CEntity*>* GetEntitiesVector() { return &entities; };
52     static const std::vector<class CComponent*>* GetOpaqueCompsVector() { return &opaqueComps; };
53     static const std::vector<class CComponent*>* GetTranslucentCompsVector() { return &translucentComps; };
54 };
```

11.70 EventSystem.h File Reference

A generic event system to allow for code to execute across the engine without direct references.

```
#include <map>
#include <vector>
#include <string>
#include <functional>
#include <algorithm>
#include "Cerberus/Core/Utility/DebugOutput/Debug.h"
```

Classes

- class [EventSystem](#)

11.70.1 Detailed Description

A generic event system to allow for code to execute across the engine without direct references.

Author

Luke Whiting

Date

Jan 2022

11.71 EventSystem.h

[Go to the documentation of this file.](#)

```
1 /*****
9 #pragma once
10 #include <map>
11 #include <vector>
12 #include <string>
13 #include <functional>
14 #include <algorithm>
15 #include "Cerberus/Core/Utility/DebugOutput/Debug.h"
16 class EventSystem
17 {
18 public:
19     // Adds a function to the event list of the specified eventID.
20     static void AddListener(std::string eventID, std::function<void()> functionToAdd);
21
22     // Triggers all functions that are listening on the specified eventID.
23     static void TriggerEvent(std::string eventID);
24
25 private:
26     static std::map<std::string, std::vector<std::function<void()>> events;
27 };
28
```

11.72 InputManager.cpp File Reference

All the functions needed for the Input Manager.

```
#include "InputManager.h"
#include <windows.h>
```

11.72.1 Detailed Description

All the functions needed for the Input Manager.

Author

Flynn Brooks

Date

May 2022

11.73 InputManager.h File Reference

Header containing all the functions and variables needed for the Input Manager.

```
#include "Cerberus\Core\Utility\Vector3.h"
```

Classes

- class [InputManager](#)

11.73.1 Detailed Description

Header containing all the functions and variables needed for the Input Manager.

Author

Flynn Brooks

Date

May 2022

11.74 InputManager.h

[Go to the documentation of this file.](#)

```
1  /*****
9  #pragma once
10 #include "Cerberus\Core\Utility\Vector3.h"
11
12 class InputManager
13 {
14 public:
15     enum Keys
16     {
17         A = 0,
18         B,
19         C,
20         D,
21         E,
22         F,
23         G,
24         H,
25         I,
26         J,
27         K,
28         L,
29         M,
30         N,
31         O,
32         P,
33         Q,
34         R,
35         S,
36         T,
37         U,
38         V,
39         W,
40         X,
41         Y,
42         Z,
43         Num0,
44         Num1,
```

```
45     Num2,
46     Num3,
47     Num4,
48     Num5,
49     Num6,
50     Num7,
51     Num8,
52     Num9,
53     Escape,
54     LControl,
55     LShift,
56     LAlt,
57     LWindows,
58     RControl,
59     RShift,
60     RAlt,
61     RWindows,
62     Menu,
63     LBracket,
64     RBracket,
65     Semicolon,
66     Comma,
67     Period,
68     Slash,
69     Backslash,
70     Tilde,
71     Equals,
72     Minus,
73     Space,
74     Enter,
75     Backspace,
76     Tab,
77     PageUp,
78     PageDown,
79     End,
80     Home,
81     Insert,
82     Delete,
83     Add,
84     Subtract,
85     Multiply,
86     Divide,
87     Left,
88     Right,
89     Up,
90     Down,
91     Numpad0,
92     Numpad1,
93     Numpad2,
94     Numpad3,
95     Numpad4,
96     Numpad5,
97     Numpad6,
98     Numpad7,
99     Numpad8,
100    Numpad9,
101    F1,
102    F2,
103    F3,
104    F4,
105    F5,
106    F6,
107    F7,
108    F8,
109    F9,
110    F10,
111    F11,
112    F12,
113    COUNT
114 };
115
116 enum Mouse
117 {
118     LButton,
119     RButton,
120     MButton,
121     MCOUNT
122 };
123
124 static Vector3 mousePos;
125
126 static bool IsKeyPressed(Keys key);
127 static bool IsKeyPressedDown(Keys key);
128 static bool IsKeyReleased(Keys key);
129 static bool IsMouseButtonPressed(Mouse mouse);
130 static bool IsMouseButtonPressedDown(Mouse mouse);
131 static bool IsMouseButtonReleased(Mouse mouse);
```

```

132
133 private:
134     static int GetKeyCode(Keys key);
135     static int GetMouseCode(Mouse mouse);
136
137     static bool keyboardKeyStates[InputManager::Keys::COUNT];
138     static bool mouseKeyStates[InputManager::Mouse::MOUNT];
139 };

```

11.75 IO.h File Reference

A Utility class to make [IO](#) easier to use.

```
#include <string>
```

Classes

- class [IO](#)

11.75.1 Detailed Description

A Utility class to make [IO](#) easier to use.

Author

Everyone

Date

May 2022

11.76 IO.h

[Go to the documentation of this file.](#)

```

1  /*****
9  #pragma once
10 #include <string>
11
12 class IO
13 {
14 public:
15
22     static std::string FindExtension(const std::string& path)
23     {
24         //store the position of last '.' in the file name
25         size_t position = path.find_last_of(".");
26         //store the characters after the '.' from the file_name string
27         if (position != -1)
28             return path.substr(position + 1);
29         else
30             return "";
31     }
32 };

```

11.77 Math.h File Reference

Utility [Math](#) Class.

```
#include "Cerberus/Core/Engine.h"
```

Classes

- class [Math](#)

Class of all the static maths functions that don't fit into existing classes.

11.77.1 Detailed Description

Utility [Math](#) Class.

Author

Everyone

Date

May 2022

11.78 Math.h

[Go to the documentation of this file.](#)

```
1  /*****  
9  #pragma once  
10  
11  #include "Cerberus/Core/Engine.h"  
12  
16  class Math  
17  {  
18  public:  
19      static int random(int min, int max);  
20  
29      static XMFLOAT3 FromScreenToWorld(const XMFLOAT3& vec);  
30  
41      static std::string FloatToStringWithDigits(const float& number, const unsigned char  
        numberOfDecimalPlaces = 3, const bool preserveDecimalZeros = false, const unsigned char  
        numberOfIntegralPlacesZeros = 1);  
42  
51      static std::string IntToString(const int& number, const unsigned char numberOfIntegralPlacesZeros =  
        1);  
52  
56      static float DegToRad(const float& degrees) { return degrees * 0.0174533f; }  
57  
61      static float RadToDeg(const float& radians) { return radians * 57.2958f; }  
62  };
```

11.79 Vector3.h

```

1  #pragma once
2
3  #include <immintrin.h>
4  #include <cmath>
5  #include <directxmath.h>
6  #include <DirectXCollision.h>
7
8  template<class T>
9  class Vector3Base
10 {
11 public:
12
13
14     #pragma warning(push)
15     //Disabled warning for 4324 since we dont care about alignment specifically. Re-Enable is alignment
    of the union becomes a problem.
16     #pragma warning(disable : 4324)
17     //Disabled warning for 4201 since having a anonymous struct is nice when using the classes
    functionality. Otherwise it would be cumbersome to use.
18     #pragma warning(disable : 4201)
19     union
20     {
21         struct { T x, y, z; };
22
23
24
25         //INTRINSIC VARIABLE, DO NOT TOUCH OR YOU WILL BE GUTTED LIKE A FISH
26         __m128 intrinsic;
27     };
28
29     #pragma warning(pop)
30
31     Vector3Base(DirectX::XMFLOAT3 Input) : intrinsic(_mm_setr_ps(Input.x, Input.y, Input.z, 0)) {}
32
33     Vector3Base() : intrinsic(_mm_setzero_ps()) {}
34
35     Vector3Base(T X, T Y, T Z) : intrinsic(_mm_setr_ps(X, Y, Z, 0.0f)) {}
36
37     Vector3Base(T AllAxis) : intrinsic(_mm_setr_ps(AllAxis, AllAxis, AllAxis, 0.0f)) {}
38
39     Vector3Base(__m128 Data) : intrinsic(Data) {}
40
41     DirectX::XMFLOAT3 ToXMFLOAT3() { return DirectX::XMFLOAT3(x, y, z); }
42
43
44     ~Vector3Base()
45     {
46         intrinsic = _mm_setzero_ps();
47     }
48
49
50
51
52
53     //
54     //FLOAT TO VECTOR
55     //
56
57
58     //Multiply with float operator
59     Vector3Base operator * (const T& OtherFloat) const { return _mm_mul_ps(intrinsic,
    _mm_set1_ps(OtherFloat)); }
60
61     //Divide with float operator
62     Vector3Base operator / (const T& OtherFloat) const { return _mm_div_ps(intrinsic,
    _mm_set1_ps(OtherFloat)); }
63
64     //Multiply with float operator
65     Vector3Base operator + (const T& OtherFloat) const { return _mm_add_ps(intrinsic,
    _mm_set1_ps(OtherFloat)); }
66
67     //Divide with float operator
68     Vector3Base operator - (const T& OtherFloat) const { return _mm_sub_ps(intrinsic,
    _mm_set1_ps(OtherFloat)); }
69
70
71
72
73     //
74     // VECTOR TO VECTOR
75     //
76
77
78
79     //Multiply vector with other vector

```

```

80     Vector3Base operator * (const Vector3Base OtherVector) const { return _mm_mul_ps(intrinsic,
OtherVector.intrinsic); }
81
82     //Minus vector with other vector
83     Vector3Base operator - (const Vector3Base OtherVector) const { return _mm_sub_ps(intrinsic,
OtherVector.intrinsic); }
84
85     //Add Vector with other vector
86     Vector3Base operator + (const Vector3Base OtherVector) const { return _mm_add_ps(intrinsic,
OtherVector.intrinsic); }
87
88     //Divide vector by other vector
89     Vector3Base operator / (const Vector3Base OtherVector) const { return _mm_div_ps(intrinsic,
OtherVector.intrinsic); }
90
91
92
93     //
94     // DIRECT OPERATORS
95     //
96
97
98     // Directly add a vector to the current vector
99     Vector3Base& operator += (const Vector3Base& OtherVector) { intrinsic = _mm_add_ps(intrinsic,
OtherVector.intrinsic); return *this; }
100     //Directly multiply the current vector by another vector
101     Vector3Base& operator *= (const Vector3Base& OtherVector) { intrinsic = _mm_mul_ps(intrinsic,
OtherVector.intrinsic); return *this; }
102     //Directly divide the vector by another vector
103     Vector3Base& operator /= (const Vector3Base& OtherVector) { intrinsic = _mm_div_ps(intrinsic,
OtherVector.intrinsic); return *this; }
104     //Directly subtract a vector from the current vector
105     Vector3Base& operator -= (const Vector3Base& OtherVector) { intrinsic = _mm_sub_ps(intrinsic,
OtherVector.intrinsic); return *this; }
106
107     //Compare and return the result of two Vector3s. return true if they are the same.
108     bool operator ==(const Vector3Base& B) const { return ((_mm_movemask_ps(_mm_cmpeq_ps(intrinsic,
B.intrinsic))) & 0x7) == 0x7; }
109
110     //Compare and return the result of two Vector3s. returns true if they are not the same.
111     bool operator !=(const Vector3Base& B) const { return ((_mm_movemask_ps(_mm_cmpeq_ps(intrinsic,
B.intrinsic))) & 0x7) != 0x7; }
112
113
114
115
116     //MATH FUNCTIONS
117
118
119
120
121     float Magnitude() const { return _mm_cvtss_f32(_mm_sqrt_ss(_mm_dp_ps(intrinsic, intrinsic, 0x71))); }
122
123
124     float Dot(const Vector3Base OtherVector) const { return _mm_cvtss_f32(_mm_dp_ps(intrinsic,
OtherVector.intrinsic, 0x71)); }
125
126     float DistanceTo(const Vector3Base B)
127     {
128         __m128 Dist = _mm_sub_ps(B.intrinsic, intrinsic);
129         return _mm_cvtss_f32(_mm_sqrt_ss(_mm_dp_ps(Dist, Dist, 0x71)));
130     }
131
132     Vector3Base& Normalize()
133     {
134         intrinsic = _mm_div_ps(intrinsic, _mm_sqrt_ps(_mm_dp_ps(intrinsic, intrinsic, 0xFF)));
135         return *this;
136     }
137
138
139
140     float Determinant(const Vector3Base OtherVector)
141     {
142         // x1 * y2 - y1 * x2;
143         //
144         _mm_cvtss_f32 _mm_sub_ps(_mm_mul_ps(intrinsic, OtherVector.intrinsic), _mm_mul_ps(intrinsic,
OtherVector.intrinsic));
145         return ((x * OtherVector.y) - (y * OtherVector.x));
146     }
147
148
149     Vector3Base Lerp(const Vector3Base A, const Vector3Base B, float Alpha)
150     {
151         return _mm_add_ps(A.intrinsic, _mm_mul_ps(_mm_sub_ps(B.intrinsic, A.intrinsic),
_mm_set1_ps(Alpha)));
152     }

```

```

153
154 void Truncate(float max)
155 {
156     if (this->Magnitude() > max)
157     {
158         this->Normalize();
159     }
160     *this *= max;
161 }
162 }
163
164
165 };
166
167
168
169
170
171 template<class T>
172 class Vector2Base
173 {
174 public:
175 #pragma warning(push)
176     //Disabled warning for 4324 since we dont care about alignment specifically. Re-Enable is alignment
    of the union becomes a problem.
177 #pragma warning(disable : 4324)
178 //Disabled warning for 4201 since having a anonymous struct is nice when using the classes
    functionality. Otherwise it would be cumbersome to use.
179 #pragma warning(disable : 4201)
180 union
181 {
182     struct { T x, y; };
183     //INTRINSIC VARIABLE, DO NOT TOUCH OR YOU WILL BE GUTTED LIKE A FISH
184     __m128 intrinsic;
185 };
186
187 Vector2Base(DirectX::XMFLOAT3 Input) : intrinsic(_mm_setr_ps(Input.x, Input.y, 0,0)) {}
188
189 Vector2Base() : intrinsic(_mm_setzero_ps()) {}
190
191 Vector2Base(T X, T Y) : intrinsic(_mm_setr_ps(X, Y, 0,0)) {}
192
193 Vector2Base(T AllAxis) : intrinsic(_mm_setr_ps(AllAxis, AllAxis, 1, 1)) {}
194
195 Vector2Base(__m128 Data) : intrinsic(Data) {}
196
197 DirectX::XMFLOAT3 ToXMFLOAT3() { return DirectX::XMFLOAT3(x, y); }
198
199
200 ~Vector2Base()
201 {
202     intrinsic = _mm_setzero_ps();
203 }
204
205
206
207
208
209 //
210 //FLOAT TO VECTOR
211 //
212
213
214 //Multiply with float operator
215 Vector2Base operator * (const T& OtherFloat) const { return _mm_mul_ps(intrinsic,
    _mm_set1_ps(OtherFloat)); }
216
217 //Divide with float operator
218 Vector2Base operator / (const T& OtherFloat) const { return _mm_div_ps(intrinsic,
    _mm_set1_ps(OtherFloat)); }
219
220 //Multiply with float operator
221 Vector2Base operator + (const T& OtherFloat) const { return _mm_add_ps(intrinsic,
    _mm_set1_ps(OtherFloat)); }
222
223 //Divide with float operator
224 Vector2Base operator - (const T& OtherFloat) const { return _mm_sub_ps(intrinsic,
    _mm_set1_ps(OtherFloat)); }
225
226
227
228
229 //
230 // VECTOR TO VECTOR
231 //
232
233

```



```

234
235 //Multiply vector with other vector
236 Vector2Base operator * (const Vector2Base OtherVector) const { return _mm_mul_ps(intrinsic,
OtherVector.intrinsic); }
237
238 //Minus vector with other vector
239 Vector2Base operator - (const Vector2Base OtherVector) const { return _mm_sub_ps(intrinsic,
OtherVector.intrinsic); }
240
241 //Add Vector with other vector
242 Vector2Base operator + (const Vector2Base OtherVector) const { return _mm_add_ps(intrinsic,
OtherVector.intrinsic); }
243
244 //Divide vector by other vector
245 Vector2Base operator / (const Vector2Base OtherVector) const { return _mm_div_ps(intrinsic,
OtherVector.intrinsic); }
246
247
248
249 //
250 // DIRECT OPERATORS
251 //
252
253
254 // Directly add a vector to the current vector
255 Vector2Base& operator += (const Vector2Base& OtherVector) { intrinsic = _mm_add_ps(intrinsic,
OtherVector.intrinsic); return *this; }
256 //Directly multiply the current vector by another vector
257 Vector2Base& operator *= (const Vector2Base& OtherVector) { intrinsic = _mm_mul_ps(intrinsic,
OtherVector.intrinsic); return *this; }
258 //Directly divide the vector by another vector
259 Vector2Base& operator /= (const Vector2Base& OtherVector) { intrinsic = _mm_div_ps(intrinsic,
OtherVector.intrinsic); return *this; }
260 //Directly subtract a vector from the current vector
261 Vector2Base& operator -= (const Vector2Base& OtherVector) { intrinsic = _mm_sub_ps(intrinsic,
OtherVector.intrinsic); return *this; }
262
263 //Compare and return the result of two Vector3s. return true if they are the same.
264 bool operator ==(const Vector2Base& B) const { return ((_mm_movemask_ps(_mm_cmpeq_ps(intrinsic,
B.intrinsic))) & 0x7) == 0x7; }
265
266 //Compare and return the result of two Vector3s. returns true if they are not the same.
267 bool operator !=(const Vector2Base& B) const { return ((_mm_movemask_ps(_mm_cmpeq_ps(intrinsic,
B.intrinsic))) & 0x7) != 0x7; }
268
269
270
271
272 //MATH FUNCTIONS
273
274
275
276
277 float Magnitude() const { return _mm_cvtss_f32(_mm_sqrt_ss(_mm_dp_ps(intrinsic, intrinsic, 0x71)));
}
278
279
280 float Dot(const Vector2Base OtherVector) const { return _mm_cvtss_f32(_mm_dp_ps(intrinsic,
OtherVector.intrinsic, 0x71)); }
281
282 float DistanceTo(const Vector2Base B)
283 {
284     __m128 Dist = _mm_sub_ps(B.intrinsic, intrinsic);
285     return _mm_cvtss_f32(_mm_sqrt_ss(_mm_dp_ps(Dist, Dist, 0x71)));
286 }
287
288 Vector2Base& Normalize()
289 {
290     intrinsic = _mm_div_ps(intrinsic, _mm_sqrt_ps(_mm_dp_ps(intrinsic, intrinsic, 0xFF)));
291     return *this;
292 }
293
294
295 float Determinant(const Vector2Base OtherVector)
296 {
297     // x1 * y2 - y1 * x2;
298     //
299     _mm_cvtss_f32 _mm_sub_ps(_mm_mul_ps(intrinsic, OtherVector.intrinsic), _mm_mul_ps(intrinsic,
OtherVector.intrinsic));
300     return ((x * OtherVector.y) - (y * OtherVector.x));
301 }
302
303
304 Vector2Base Lerp(const Vector2Base A, const Vector2Base B, float Alpha)
305 {
306     return _mm_add_ps(A.intrinsic, _mm_mul_ps(_mm_sub_ps(B.intrinsic, A.intrinsic),
_mm_set1_ps(Alpha)));

```

```

307     }
308
309
310
311     void Truncate(float max)
312     {
313         if (this->Magnitude() > max)
314         {
315             this->normalize();
316
317             *this *= max;
318         }
319     }
320
321 };
322
323
324
325 typedef Vector3Base<unsigned int> Vector3I;
326
327 typedef Vector3Base<float> Vector3;
328
329
330 typedef Vector2Base<unsigned int> Vector2I;
331
332 typedef Vector2Base<float> Vector2;
333
334
335
336
337
338
339
340 //0.025000
341 //0.025000
342
343

```

11.80 Cerberus/Resource.h

```

1  //{NO_DEPENDENCIES}
2  // Microsoft Visual C++ generated include file.
3  // Used by Tutorial01.rc
4  //
5  #define IDC_MYICON                2
6  #define IDD_TUTORIAL1_DIALOG      102
7  #define IDS_APP_TITLE             103
8  #define IDD_ABOUTBOX              103
9  #define IDM_ABOUT                 104
10 #define IDM_EXIT                  105
11 #define IDI_SMALL                  108
12 #define IDC_TUTORIAL1             109
13 #define IDR_MAINFRAME              128
14 #define IDI_ICON1                  129
15 #define IDI_ICON2                  131
16 #define IDC_STATIC                 -1
17
18 // Next default values for new objects
19 //
20 #ifdef APSTUDIO_INVOKED
21 #ifndef APSTUDIO_READONLY_SYMBOLS
22 #define _APS_NO_MFC                1
23 #define _APS_NEXT_RESOURCE_VALUE    132
24 #define _APS_NEXT_COMMAND_VALUE    32771
25 #define _APS_NEXT_CONTROL_VALUE    1000
26 #define _APS_NEXT_SYMED_VALUE      110
27 #endif
28 #endif

```

11.81 Necrodoggiecon/Resource.h

```

1  //{NO_DEPENDENCIES}
2  // Microsoft Visual C++ generated include file.
3  // Used by Tutorial01.rc
4  //
5  #define IDC_MYICON                2
6  #define IDD_TUTORIAL1_DIALOG      102
7  #define IDS_APP_TITLE             103
8  #define IDD_ABOUTBOX              103

```

```

9 #define IDM_ABOUT 104
10 #define IDM_EXIT 105
11 #define IDI_TUTORIAL1 107
12 #define IDI_SMALL 108
13 #define IDC_TUTORIAL1 109
14 #define IDR_MAINFRAME 128
15 #define IDI_ICON1 129
16 #define IDI_ICON2 131
17 #define IDC_STATIC -1
18
19 // Next default values for new objects
20 //
21 #ifdef APSTUDIO_INVOKED
22 #ifndef APSTUDIO_READONLY_SYMBOLS
23 #define _APS_NO_MFC 1
24 #define _APS_NEXT_RESOURCE_VALUE 132
25 #define _APS_NEXT_COMMAND_VALUE 32771
26 #define _APS_NEXT_CONTROL_VALUE 1000
27 #define _APS_NEXT_SYMED_VALUE 110
28 #endif
29 #endif

```

11.82 CT_EditorEntity.h

```

1 #pragma once
2 #include "Cerberus\Core\CEntity.h"
3
4 enum class EditorEntityType
5 {
6     None, Standard, Enemy, Interactable, Waypoint, Flag, WeaponHolder
7 };
8 class CT_EditorEntity :
9     public CEntity
10 {
11 protected:
12
13     // class CSpriteComponent* sprite = nullptr;
14
15     int entitySlotID;
16
17     EditorEntityType inspectType;
18
19 public:
20
21     class CSpriteComponent* sprite = nullptr;
22
23     CT_EditorEntity();
24
25     virtual void Update(float deltaTime) override;
26
27     virtual void InitialiseEntity(int SlotID);
28
29     // virtual void SaveEntity(int Index, int MapSlot);
30
31     EditorEntityType GetType() { return inspectType; }
32
33     int GetSlot() { return entitySlotID; }
34
35
36
37
38
39 };
40
41 class CT_EditorEntity_WeaponHolder :
42     public CT_EditorEntity
43 {
44 protected:
45
46     // class CSpriteComponent* sprite = nullptr;
47
48
49     char* current_item = (char*)"Dagger";
50     int itemSlot = 0;
51
52     CSpriteComponent* weaponSprite;
53 public:
54
55
56
57     CT_EditorEntity_WeaponHolder();
58
59

```

```

60     char* GetWeaponName() { return current_item; }
61     int GetAssignedWeapon() { return itemSlot; }
62     void AssignWeapon(char* WeaponID, int Index);
63
64     virtual void Update(float deltaTime) override;
65
66
67
68     virtual void InitialiseEntity(int SlotID);
69
70
71
72 };
73
74
75
76 class CT_EditorEntity_Waypoint :
77     public CT_EditorEntity
78 {
79 protected:
80
81     // class CSpriteComponent* sprite = nullptr;
82
83
84
85
86
87 public:
88
89     Vector2 GetGridPos();
90
91     CT_EditorEntity_Waypoint();
92
93
94     int waypointOrder;
95     Vector2 gridPos;
96
97     virtual void Update(float deltaTime) override;
98
99
100
101
102     virtual void InitialiseEntity(int SlotID);
103
104
105
106 };
107
108
109 class CT_EditorEntity_Enemy :
110     public CT_EditorEntity
111 {
112 protected:
113
114     // class CSpriteComponent* sprite = nullptr;
115
116     bool displayWaypoints = false;
117
118     char* current_item = (char*)"Dagger";
119     int itemIndex = 0;
120
121     float health = 2.0f;
122     float speed = 100.0f;
123
124     float mass = 10.0f;
125     float range = 200.0f;
126     float viewAngle = 90.0f;
127
128     float rotationSpeed = 0.01f;
129     float maxSearchTime = 5.0f;
130
131     bool isBoss = false;
132
133 public:
134
135     float GetHealth() { return health; }
136     float GetSpeed() { return speed; }
137     float GetMass() { return mass; }
138     float GetRange() { return range; }
139     float GetViewAngle() { return viewAngle; }
140     float GetRotationSpeed() { return rotationSpeed; }
141     float GetMaxSearchTime() { return maxSearchTime; }
142     bool GetIsBoss() { return isBoss; }
143
144     void SetHealth(float newHealth) { health = newHealth; }
145     void SetSpeed(float newSpeed) { speed = newSpeed; }
146     void SetMass(float newMass) { mass = newMass; }

```

```

147 void SetRange(float newRange) { range = newRange; }
148 void SetViewAngle(float newViewAngle) { viewAngle = newViewAngle; }
149 void SetRotationSpeed(float newRotationSpeed) { rotationSpeed = newRotationSpeed; }
150 void SetMaxSearchTime(float newMaxSearchTime) { maxSearchTime = newMaxSearchTime; }
151 void SetIsBoss(bool newIsBoss) { isBoss = newIsBoss; }
152 std::vector<CT_EditorEntity_Waypoint*> Waypoints;
153
154 char* GetWeaponName() { return current_item; }
155 int GetAssignedWeapon() { return itemIndex; }
156 void AssignWeapon(char* WeaponID, int Index);
157
158 CT_EditorEntity_Enemy();
159
160 virtual void Update(float deltaTime) override;
161
162
163 virtual void InitialiseEntity(int SlotID);
164
165
166
167 void ToggleWaypoints(bool Display);
168
169 CT_EditorEntity_Waypoint* AddWaypoint(Vector2 Position);
170
171 void RemoveWaypoint(int Index);
172
173
174
175
176
177
178
179
180 };
181
182 class CT_EditorEntity_PlayerStart :
183     public CT_EditorEntity
184 {
185 public:
186
187     CT_EditorEntity_PlayerStart();
188
189     virtual void Update(float deltaTime) override;
190
191
192
193
194
195
196 };
197
198

```

11.83 CT_EditorGrid.h

```

1 #pragma once
2 #include "Cerberus\Core\CEntity.h"
3 #include "Cerberus\Core\Environment\CWorld_Edit.h"
4
5 class CT_EditorGrid :
6     public CEntity
7 {
8 public:
9     CT_EditorGrid();
10
11     virtual void Update(float deltaTime) override;
12
13     void SetupGrid();
14
15
16     ~CT_EditorGrid();
17
18     class CGridCursor* cursorEntity;
19
20
21
22     void SetupGrid(class CCameraComponent* cam);
23
24 protected:
25     class CSpriteComponent* gridSprite = nullptr;
26
27 };
28

```

11.84 CT_EditorMain.h

```

1 #pragma once
2 #include "Cerberus\Core\CEntity.h"
3 class CT_EditorMain
4 {
5 public:
6     CT_EditorMain();
7
8     void Initialise();
9
10    ~CT_EditorMain();
11
12    void RenderWindows();
13
14    class CT_EditorGrid* grid;
15
16    class CT_EditorWindows* editorWindow;
17
18 };
19
20
21
22

```

11.85 CT_EditorWindows.h

```

1 #pragma once
2
3 #include "Dependencies/IMGUI/imgui.h"
4 #include "Dependencies/IMGUI/imgui_impl_dx11.h"
5 #include "Dependencies/IMGUI/imgui_impl_win32.h"
6
7 #include <corecrt_malloc.h>
8 #include <iostream>
9 #include "Cerberus\Core\Utility\Vector3.h"
10 #include <vector>
11
12 class CT_EditorWindows
13 {
14
15     char InputBuf[256];
16     ImVector<char*> Items;
17     ImVector<const char*> Commands;
18     ImVector<char*> History;
19     int HistoryPos; // -1: new line, 0..History.Size-1 browsing history.
20     ImGuiTextFilter Filter;
21     bool AutoScroll;
22     bool ScrollToBottom;
23
24     bool* open;
25     int* levelToLoad;
26     bool toggleWaypoints;
27     const char* weaponNames[9] = {};
28     std::vector<std::string> WepList;
29
30 protected:
31
32     const char* WindowTitle = "Editor Window";
33     Vector2 WindowScale = (256.0f, 256.0f);
34
35 public:
36     CT_EditorWindows()
37     {
38         ClearLog();
39         memset(InputBuf, 0, sizeof(InputBuf));
40         HistoryPos = -1;
41
42         AutoScroll = true;
43         ScrollToBottom = false;
44         open = new bool(true);
45         //levelToLoad = new int(0);
46         toggleWaypoints = false;
47         LoadWeapons();
48         InitialiseMapSlot();
49     }
50
51     ~CT_EditorWindows()
52     {
53         ClearLog();
54         for (int i = 0; i < History.Size; i++)
55             free(History[i]);
56     }
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

```

57
58 private:
59
60     // Portable helpers
61     static int Stricmp(const char* s1, const char* s2) { int d; while ((d = toupper(*s2) -
62     toupper(*s1)) == 0 && *s1) { s1++; s2++; } return d; }
63     static int Strnicmp(const char* s1, const char* s2, int n) { int d = 0; while (n > 0 && (d =
64     toupper(*s2) - toupper(*s1)) == 0 && *s1) { s1++; s2++; n--; } return d; }
65     static char* Strdup(const char* s) { IM_ASSERT(s); size_t len = strlen(s) + 1; void* buf =
66     malloc(len); IM_ASSERT(buf); return (char*)memcpy(buf, (const void*)s, len); }
67     static void Strtrim(char* s) { char* str_end = s + strlen(s); while (str_end > s && str_end[-1] == '
68     ') str_end--; *str_end = 0; }
69
70     bool debugModeToggle = false;
71
72 public:
73
74     void ClearLog()
75     {
76         for (int i = 0; i < Items.Size; i++)
77             free(Items[i]);
78         Items.clear();
79     }
80
81     // Use [error] to define errors.
82     void AddLog(const char* fmt, ...) IM_FMTARGS(2)
83     {
84         // FIXME-OPT
85         char buf[1024];
86         va_list args;
87         va_start(args, fmt);
88         vsnprintf(buf, IM_ARRAYSIZE(buf), fmt, args);
89         buf[IM_ARRAYSIZE(buf) - 1] = 0;
90         va_end(args);
91         Items.push_back(Strdup(buf));
92     }
93
94     void LoadWeapons();
95     void InitialiseMapSlot();
96
97     void render();
98
99 };

```

11.86 WorldConstants.h

```

1 #pragma once
2
3 enum class EntityType
4 {
5     Player,
6     MeleeCharacter,
7     RangedCharacter,
8     misc
9 };
10
11
12 enum class CellType
13 {
14     Empty,
15     Edge,
16     Floor,
17     OuterCorner,
18     InnerCorner,
19     TConnector,
20     XConnector
21 };
22
23 enum class CellID
24 {
25     N = 0,
26     F = 1,
27     W_N = 2,
28     W_E = 3,
29     W_S = 4,
30     W_W = 5,
31     IC_NW = 6,
32     IC_NE = 7,
33     IC_SW = 8,
34     IC_SE = 9,

```

```

35     OC_NW = 10,
36     OC_NE = 11,
37     OC_SW = 12,
38     OC_SE = 13,
39
40
41     W_T = 13,
42     C_TR = 14,
43     C_TL = 15,
44
45
46     WC_HS = 16,
47     WC_HN = 17,
48     WC_VE = 18,
49     WC_VW = 19,
50
51
52 };
53
54 struct CT_PropData
55 {
56     CT_PropData(int ID, int Coordinate)
57     {
58         propID = ID;
59         coordinate = Coordinate;
60     }
61     int propID;
62     Vector3 coordinate;
63 };
64
65
66 #define tileSize 32
67 #define mapScale 64
68 #define tileSizeMultiplier 2

```

11.87 CerberusTools/CursorEntity.h

```

1 #pragma once
2
3 #include "Cerberus/Core/CEntity.h"
4
5
6 class CursorEntity : public CEntity
7 {
8     class CAnimationSpriteComponent* sprite = nullptr;
9     class CTextRenderComponent* text = nullptr;
10     float timeElapsed = 0;
11
12     Vector3 mouseOffset = { 0,0,0 };
13     bool mouseRHeld = false;
14
15 public:
16     CursorEntity();
17     virtual void Update(float deltaTime) override;
18     virtual ~CursorEntity();
19 };
20

```

11.88 Necrodoggiecon/Game/CursorEntity.h

```

1 #pragma once
2 #include "Cerberus\Core\CEntity.h"
3
4 class CursorEntity : public CEntity
5 {
6     class CAnimationSpriteComponent* sprite = nullptr;
7     class CTextRenderComponent* text = nullptr;
8     float timeElapsed = 0;
9
10     Vector3 mouseOffset = { 0,0,0 };
11     bool mouseRHeld = false;
12
13 public:
14     CursorEntity();
15     virtual void Update(float deltaTime) override;
16     virtual ~CursorEntity();
17 };
18

```


11.89 CWorld_Game.h

```
1 #pragma once
2 #include "Cerberus\Core\Environment\CWorld.h"
3 class CWorld_Game :
4     public CWorld
5 {
6
7
8
9 public:
10
11     CWorld_Game(int Slot);
12
13
14
15     virtual void SetupWorld();
16
17     virtual void UnloadWorld();
18
19     virtual void ReloadWorld();
20
21     virtual void LoadEnemyUnits(int Slot);
22     virtual void LoadEntities(int Slot) override;
23
24 };
25
```

11.90 CWorld_Menu.h

```
1 #pragma once
2 #include "Cerberus\Core\Environment\CWorld.h"
3 class CWorld_Menu :
4     public CWorld
5 {
6
7     virtual void SetupWorld() override;
8
9 };
10
```

11.91 AlarmEnemy.cpp File Reference

File containing all the functions needed for the alarm enemy.

```
#include "AlarmEnemy.h"
#include "Game/SoundManager.h"
```

11.91.1 Detailed Description

File containing all the functions needed for the alarm enemy.

Author

Nasser Ksous

Date

May 2022

11.92 AlarmEnemy.h File Reference

Header file for the alarm enemy.

```
#include "Necrodoggiecon\Game\AI\CAIController.h"
```

Classes

- class [AlarmEnemy](#)
Class for the alarm enemy.

11.92.1 Detailed Description

Header file for the alarm enemy.

Author

Nasser Ksous

Date

May 2022

11.93 AlarmEnemy.h

[Go to the documentation of this file.](#)

```
1 /*****/
8 #pragma once
9 #include "Necrodoggiecon\Game\AI\CAIController.h"
10
14 class AlarmEnemy :
15     public CAIController
16 {
17 public:
18     AlarmEnemy();
19
20     virtual void Update(float deltaTime) override;
21     virtual void ChasePlayer(CCharacter* player) override;
22
23 protected:
24     virtual void OnDeath() override;
25     virtual void OnHit(const std::string& hitSound) override;
26
27 private:
28     float alarmTimer = 10.0f;
29     bool onCooldown = false;
30 };
31
```

11.94 CAIController.cpp File Reference

All the functions needed to control the AI.

```
#include "CAIController.h"
#include "Cerberus/Core/Utility/CWorldManager.h"
#include "Cerberus\Core\Environment\CWorld.h"
#include "Game/NecrodoggieconPage.h"
```

11.94.1 Detailed Description

All the functions needed to control the AI.

Author

Nasser Ksous

Date

May 2022

11.95 CAIController.h File Reference

Header file containing all the functions and variables needed to control the AI.

```
#include <iostream>
#include "Cerberus\Core\CEntity.h"
#include "Cerberus\Core\Utility\Vector3.h"
#include "Cerberus\Core\Components\CSpriteComponent.h"
#include "Cerberus\Core\Utility\EventSystem\EventSystem.h"
#include "Cerberus\Core\Engine.h"
#include "Cerberus\Core\Utility\Audio\AudioController.h"
#include "Cerberus\Core\Components\CAudioEmitterComponent.h"
#include "Necrodoggiecon\Game\AI\State.h"
#include "Cerberus\Core\AI\Pathfinding.h"
#include "Necrodoggiecon\Game\CCharacter.h"
```

Classes

- class [CAIController](#)
Controller class for the AI.

11.95.1 Detailed Description

Header file containing all the functions and variables needed to control the AI.

Author

Nasser Ksous

Date

May 2022

11.96 CAIController.h

[Go to the documentation of this file.](#)

```

1 #pragma once
2 /*****
10 #include <iostream>
11 #include "Cerberus\Core\CEntity.h"
12 #include "Cerberus\Core\Utility\Vector3.h"
13 #include "Cerberus\Core\Components\CSpriteComponent.h"
14 #include "Cerberus\Core\Utility\EventSystem\EventSystem.h"
15 #include "Cerberus\Core\Engine.h"
16 #include "Cerberus\Core\Utility\Audio\AudioController.h"
17 #include "Cerberus\Core\Components\CAudioEmitterComponent.h"
18
19 #include "Necrodoggiecon/Game/AI/State.h"
20 #include "Cerberus/Core/AI/Pathfinding.h"
21 #include "Necrodoggiecon\Game\CCharacter.h"
22
26 class CAIController : public CCharacter
27 {
28 public:
29     CAIController();
30     ~CAIController();
31
32     void SetRotationSpeed(float speed);
33     float GetRotationSpeed();
34
35     void SetSearchTime(float time);
36     float GetSearchTime();
37
38     void SetInitialSpeed(float speed);
39     float GetInitialSpeed();
40     void SetSpeed(float speed);
41     float GetSpeed();
42     void SetMass(float mass);
43     float GetMass();
44     void SetRange(float range);
45     float GetRange();
46     void SetViewAngle(float angle);
47     float GetViewAngle();
48
49     void SetWidth(float wide);
50     float GetWidth();
51     void SetHeight(float high);
52     float GetHeight();
53
54     void SetPositionToInvestigate(Vector3 pos);
55     Vector3 GetPositionToInvestigate();
56
57     void SetIsAttacking(bool isAttack);
58     bool GetIsAttacking();
59
60     void SetSpriteSize(float size);
61     float GetSpriteSize();
62
63     void SetIsBoss(bool boss);
64     bool GetIsBoss();
65
66     virtual void Update(float deltaTime) override;
67
68     void Patrolling();
69     void SearchForPlayer();
70     void Investigating(Vector3 positionOfInterest);
71
72     virtual void AttackEnter(CCharacter* player);
73     virtual void ChaseEnter();
74     virtual void ChasePlayer(CCharacter* player);
75     virtual void AttackPlayer(CCharacter* player, float deltaTime);
76
77     void SetCurrentState(State& state);
78     bool CanSee(CCharacter* player);
79
80     void SetPathNodes(std::vector<WaypointNode*> nodes);
81     Pathfinding* pathing;
82     void SetPath();
83     void SetPath(Vector3 endPosition);
84
85     void ApplyDamage(float damageAmount);
86     void ApplyDamage(float damageAmount, const std::string& hitAudioPath);
87
88     class CAnimationSpriteComponent* sprite = nullptr;
89
90 protected:
91     virtual void OnHit(const std::string& hitSound) {};
92     virtual void OnDeath() {};

```

```

93
94     class CSpriteComponent* viewFrustrum = nullptr;
95
96     Vector3 positionToInvestigate;
97     void Movement(float deltaTime);
98
99     Vector3 CollisionAvoidance();
100
101     Vector3 velocity;
102     Vector3 acceleration;
103     Vector3 heading;
104     Vector3 aiPosition;
105
106     std::vector<CTile*> tiles;
107     std::vector<CTile*> obstacles;
108
109     PatrolNode* currentPatrolNode;
110
111     std::vector<WaypointNode*> pathNodes;
112
113     Vector3 Seek(Vector3 TargetPos);
114
115     void CheckForPlayer();
116
117     void MoveViewFrustrum();
118
119     int currentCount;
120     bool isAttacking = false;
121     bool isBoss = false;
122
123     CCharacter* playerToKill = nullptr;
124     CCharacter* playerToChase = nullptr;
125
126     Vector3 originalViewFrustrumPosition;
127
128     std::vector<CCharacter*> characters = Engine::GetEntityOfType<CCharacter>();
129     std::vector<CCharacter*> players;
130
131     float aiSpeed = 100.0f;
132     float initialSpeed = aiSpeed;
133     float aiMass = 10.0f;
134     float aiRange = 400.0f;
135     float aiViewAngle = 90.0f;
136
137     float width = 64.0f;
138     float height = 64.0f;
139
140     float rotationSpeed = 0.01f;
141     float maxSearchTime = 5.0f;
142
143     float searchTimer = 0.0f;
144
145     float sizeOfTiles = 0.0f;
146
147     float spriteSize = 64.0f;
148
149     State* currentState;
150
151     virtual void HasCollided(CollisionComponent* collidedObject)
152     {
153         if (collidedObject->GetName() == "Wall")
154         {
155             colComponent->Resolve(collidedObject);
156             this->SetPosition(colComponent->GetPosition());
157         }
158     }
159 };
160

```

11.97 DogEnemy.cpp File Reference

File containing all the functions needed for the dog enemy.

```

#include "DogEnemy.h"
#include "Game/SoundManager.h"

```

11.97.1 Detailed Description

File containing all the functions needed for the dog enemy.

Author

Nasser Ksous

Date

May 2022

11.98 DogEnemy.h File Reference

Header for the dog enemy type.

```
#include "Necrodoggiecon\Game\AI\CAIController.h"
```

Classes

- class [DogEnemy](#)
Class for the dog enemy.

11.98.1 Detailed Description

Header for the dog enemy type.

Author

Nasser Ksous

Date

May 2022

11.99 DogEnemy.h

[Go to the documentation of this file.](#)

```
1 /*****/
2 #pragma once
3 #include "Necrodoggiecon\Game\AI\CAIController.h"
4
5 class DogEnemy :
6     public CAIController
7 {
8 public:
9     DogEnemy();
10
11     virtual void Update(float deltaTime) override;
12     virtual void ChasePlayer(CCharacter* player) override;
13     virtual void AttackEnter(CCharacter* player) override;
14     virtual void AttackPlayer(CCharacter* player, float deltaTime) override;
15 protected:
16     virtual void OnDeath() override;
17     virtual void OnHit(const std::string& hitSound) override;
18
19 private:
20     bool onCooldown = false;
21     float attackCooldown = 0.0f;
22     float attackTimer = 1.0f;
23     float attackRange = 300.0f;
24     const float walkAnimationSpeed = 1.3f;
25     Vector3 targetPosition;
26 };
27
28
29
30
31
32
33
34
35
36
37
38
```

11.100 GruntEnemy.cpp File Reference

All the functions needed to control the Melee Enemies.

```
#include "GruntEnemy.h"  
#include "Game/SoundManager.h"  
#include "Cerberus/Core/Utility/IO.h"
```

11.100.1 Detailed Description

All the functions needed to control the Melee Enemies.

Author

Nasser Ksous

Date

May 2022

11.101 GruntEnemy.h File Reference

Header file containing all the inherited functions from [CAIController](#) and variables needed to control the Melee Enemies.

```
#include "Necrodoggiecon\Game\AI\CAIController.h"  
#include <Necrodoggiecon/Game/WeaponInterface.h>  
#include <Necrodoggiecon/Weapons/Ranged/Crossbow.h>
```

Classes

- class [GruntEnemy](#)
Class for the Grunt enemy.

11.101.1 Detailed Description

Header file containing all the inherited functions from [CAIController](#) and variables needed to control the Melee Enemies.

Author

Nasser Ksous

Date

May 2022

11.102 GruntEnemy.h

[Go to the documentation of this file.](#)

```

1 #pragma once
2 /*****
10 #include "Necrodoggiecon\Game\AI\CAIController.h"
11 #include <Necrodoggiecon/Game/WeaponInterface.h>
12 #include <Necrodoggiecon/Weapons/Ranged/Crossbow.h>
13
17 class GruntEnemy :
18     public CAIController
19 {
20 public:
21     GruntEnemy();
22
23     virtual void ChasePlayer(CCharacter* player) override;
24     virtual void AttackPlayer(CCharacter* player, float deltaTime) override;
25 protected:
26     virtual void OnDeath() override;
27     virtual void OnHit(const std::string& hitSound) override;
28
29     virtual void Update(float deltaTime) override;
30
31     void UpdateWeaponSprite();
32 };
33

```

11.103 State.cpp File Reference

Functions for all the functions for the states.

```

#include "State.h"
#include "Necrodoggiecon\Game\AI\CAIController.h"

```

11.103.1 Detailed Description

Functions for all the functions for the states.

Author

Nasser Ksous

Date

May 2022

11.104 State.h File Reference

Header files containing the base state class and any inherited states for the FSM of the AI.

```

#include "Necrodoggiecon/Game/CCharacter.h"

```


Classes

- class [State](#)
Base state class.
- class [ChaseState](#)
State for when the AI is chasing the player.
- class [AttackState](#)
State for when the AI is attacking the player.
- class [PatrolState](#)
State for when the AI is patrolling between the patrol points.
- class [SearchState](#)
State for when the AI is searching for the player.
- class [InvestigateState](#)
State for when the AI is investigating.

11.104.1 Detailed Description

Header files containing the base state class and any inherited states for the FSM of the AI.

Author

Nasser Ksous

Date

May 2022

11.105 State.h

[Go to the documentation of this file.](#)

```

1 #pragma once
2 /*****
3  * Necrodoggiecon/Game/CCharacter.h
4  */
5 #include "Necrodoggiecon/Game/CCharacter.h"
6
7 class CAIController;
8
9 //Reference:
10 https://www.aleksandrhovhannisyan.com/blog/finite-state-machine-fsm-tutorial-implementing-an-fsm-in-c/
11
12 class State
13 {
14 public:
15     virtual void Enter(CAIController* controller) { UNREFERENCED_PARAMETER(controller); };
16     virtual void Exit(CAIController* controller) { UNREFERENCED_PARAMETER(controller); };
17     virtual void Update(CAIController* controller, float deltaTime) { UNREFERENCED_PARAMETER(controller);
18         UNREFERENCED_PARAMETER(deltaTime); };
19 };
20
21 class ChaseState : public State
22 {
23 public:
24     void Enter(CAIController* controller) override;
25     void Update(CAIController* controller, float deltaTime) override;
26     void Exit(CAIController* controller) override;
27
28     static State& getInstance();
29 private:
30     CCharacter* closestPlayer;
31 };
32
33
34
35
36
37
38
39
40
41
42

```

```

46 class AttackState : public State
47 {
48 public:
49     void Enter(CAIController* controller) override;
50     void Update(CAIController* controller, float deltaTime) override;
51     void Exit(CAIController* controller) override;
52
53     static State& getInstance();
54
55 private:
56     CCharacter* closestPlayer;
57 };
58
62 class PatrolState : public State
63 {
64 public:
65     void Enter(CAIController* controller) override;
66     void Update(CAIController* controller, float deltaTime) override;
67     void Exit(CAIController* controller) override;
68
69     static State& getInstance();
70 };
71
75 class SearchState : public State
76 {
77 public:
78     void Enter(CAIController* controller) override;
79     void Update(CAIController* controller, float deltaTime) override;
80     void Exit(CAIController* controller) override;
81
82     static State& getInstance();
83
84 private:
85     float searchTimer;
86     std::vector<CCharacter*> characters;
87     std::vector<CCharacter*> players;
88 };
89
93 class InvestigateState : public State
94 {
95 public:
96     void Enter(CAIController* controller) override;
97     void Update(CAIController* controller, float deltaTime) override;
98     void Exit(CAIController* controller) override;
99
100     static State& getInstance();
101
102 private:
103
104 };

```

11.106 AudioEmitterEntity.cpp File Reference

An entity that contains an audio emitter.

```
#include "AudioEmitterEntity.h"
```

11.106.1 Detailed Description

An entity that contains an audio emitter.

Used in the [SoundManager](#) to enable the playing of audio at specific positions.

Author

Cathan Bertram

Date

May 2022

11.107 AudioEmitterEntity.h

```

1 #pragma once
2 #include "Cerberus\Core\CEntity.h"
3 #include <Cerberus/Core/Components/CAudioEmitterComponent.h>
4 class AudioEmitterEntity :
5     public CEntity
6 {
7 public:
8     AudioEmitterEntity();
9     ~AudioEmitterEntity();
10
11     void SetAudio(const std::string& audioPath, float range);
12     void SetAudio(const std::string& audioPath, float range, bool ambient);
13     void PlayAudio(Vector3 position);
14     void Stop();
15     void PlayAudio(const std::string& audioPath);
16     void PlayAudio(bool shouldLoop);
17     void Load(const std::string& audioPath, bool ambient);
18     void SetRange(float range);
19     void SetAttachedEntity(CEntity* entity) { isAttached = true; attachedEntity = entity; }
20 protected:
21     CAudioEmitterComponent* audioEmitter;
22     CEntity* attachedEntity;
23     bool isAttached;
24     // Inherited via CEntity
25     virtual void Update(float deltaTime) override;
26 };
27

```

11.108 CCharacter.cpp File Reference

Base class for Characters.

```

#include "CCharacter.h"
#include "Necrodoggiecon\Game\WeaponPickup.h"

```

11.108.1 Detailed Description

Base class for Characters.

Author

Cathan Bertram

Date

May 2022

11.109 CCharacter.h

```

1 #pragma once
2 #include <Cerberus\Core\Components\CAAnimationSpriteComponent.h>
3 #include <Cerberus\Core\CEntity.h>
4 #include "WeaponInterface.h"
5
6 class CCharacter : public CEntity
7 {
8 private:
9 protected:
10     bool isPlayer = false;
11     bool visible = true;
12     float health = 1.0f;
13     WeaponInterface* weaponComponent = nullptr;

```

```

14     CSpriteComponent* weaponSprite = nullptr;
15
16     void UpdateWeaponSpritePosition(CSpriteComponent* wSprite);
17
18     void AddMovement(XMFLOAT2 vel, float deltaTime);
19
20
21 public:
22     virtual void ApplyDamage(float damageAmount) {};
23     virtual void ApplyDamage(float damageAmount, const std::string& onHitSound) {};
24
25     virtual void Update(float deltaTime) {};
26
27     CCharacter();
28     virtual ~CCharacter();
29
30     void EquipWeapon(Weapon* weapon);
31
32     void UpdateWeaponSprite();
33
34     void SetHealth(float heal);
35     float GetHealth();
36
37     void SetIsPlayer(bool player);
38     bool GetIsPlayer();
39
40     bool GetVisible() { return visible; }
41
42     Weapon* GetWeapon() { return weaponComponent->GetCurrentWeapon(); };
43 };
44
45
46
47

```

11.110 CInteractable.h File Reference

Entity that can be interacted with.

```

#include "Cerberus\Core\CEntity.h"
#include "Cerberus\Core\Components\CSpriteComponent.h"
#include "Cerberus\Core\Components\CTextRenderComponent.h"

```

Classes

- class [CInteractable](#)

11.110.1 Detailed Description

Entity that can be interacted with.

Acts as a base class for any entities that wish to be interacted with in specific ways.

Author

Luke Whiting

Date

May 2022

11.111 CInteractable.h

[Go to the documentation of this file.](#)

```
1 /*****
2 3 #pragma once
4 5 #include "Cerberus\Core\CEntity.h"
6 7 #include "Cerberus\Core\Components\CSpriteComponent.h"
8 9 #include "Cerberus\Core\Components\CTextRenderComponent.h"
10 11 class CInteractable : public CEntity
12 13 {
14 14 public:
15 15     CInteractable();
16 16     virtual ~CInteractable();
17 17
18 18     virtual void Update(float deltaTime) override;
19 19
20 20     virtual void OnInteract();
21 21     virtual void OnEnterOverlap();
22 22     virtual void OnLeaveOverlap();
23 23
24 24     virtual void HasCollided(CollisionComponent* collidedObject) override;
25 25
26 26     void SetTexture(std::string path);
27 27     void SetTextureWIC(std::string path);
28 28
29 29     void SetInteractRange(const float value);
30 30
31 31 protected:
32 32     void DrawUI();
33 33     CollisionComponent* GetLastCollidedObject();
34 34     CSpriteComponent* GetSprite();
35 35
36 36 private:
37 37     float interactTextOffset;
38 38     float interactRange;
39 39     CollisionComponent* lastCollidedObject;
40 40
41 41     CSpriteComponent* sprite;
42 42     CTextRenderComponent* interactText;
43 43
44 44 };
45 45
```

11.112 CPlayer.h

```
1 #pragma once
2 #include "Cerberus\Core\Engine.h"
3 #include "Cerberus\Core\CEntity.h"
4 #include <stdio.h>
5
6
7 class CPlayer : public CEntity
8 {
9     class CSpriteComponent* sprite = nullptr;
10     float timeElapsed = 0;
11 public:
12     CPlayer();
13     virtual void Update(float deltaTime) override;
14     virtual ~CPlayer();
15 };
16
```

11.113 CPlayerController.cpp File Reference

Base class for PlayerControllers, handles functionality for possessing and unpossessing characters.

```
#include "CPlayerController.h"
```

11.113.1 Detailed Description

Base class for PlayerControllers, handles functionality for possessing and unpossessing characters.

Author

Cathan Bertram

Date

May 2022

11.114 CPlayerController.h

```
1 #pragma once
2 #include <Cerberus\Core\CEntity.h>
3
4 class CCharacter;
5
6 class CPlayerController : public CEntity
7 {
8 private:
9     CCharacter* possessedCharacter = nullptr;
10     bool hasCharacter = false;
11
12 protected:
13     CCharacter* GetCharacter() { return possessedCharacter; }
14     bool HasCharacter() { return hasCharacter; }
15
16     virtual void HandleInput(float deltaTime);
17
18     virtual void OnPossess() {};
19     virtual void OnUnpossess() {};
20
21 public:
22     CPlayerController();
23     ~CPlayerController();
24
25     void Possess(CCharacter* characterToPossess);
26     void Unpossess();
27
28 };
29
30
```

11.115 Dialogue.h

```
1 #pragma once
2
3 struct Dialogue
4 {
5 public:
6     std::string name;
7     std::string dialogue;
8
9     Dialogue(std::string name, std::string dialogue) : dialogue(dialogue), name(name)
10     {
11     }
12 };
13
14
```

11.116 DialogueHandler.cpp File Reference

Static class used to control dialogue, including the loading of dialogue from a json.

```
#include "DialogueHandler.h"
#include "Cerberus/Core/Engine.h"
#include <Game/DialogueUI.h>
#include <fstream>
#include "Cerberus\Dependencies\NlohmannJson\json.hpp"
#include <Cerberus/Core/Utility/EventSystem/EventSystem.h>
```

11.116.1 Detailed Description

Static class used to control dialogue, including the loading of dialogue from a json.

Author

Cathan Bertram

Date

May 2022

11.117 DialogueHandler.h

```
1 #pragma once
2 #include <Cerberus/Core/CEntity.h>
3 #include <Game/DialogueUI.h>
4 #include <Game/Dialogue.h>
5
6 class DialogueHandler : public CEntity
7 {
8 private:
9     static DialogueUI* dialogueUI;
10    static std::vector<Dialogue*> currentDialogue;
11    static int curDialogueIndex;
12    static bool instantDisplay;
13 public:
14    DialogueHandler();
15    ~DialogueHandler();
16    static void SetDialogue(const std::string& name, const std::string& dialogue);
17    static void LoadDialogue(const std::string& jsonPath, const std::string& dialogueName);
18    static void AdvanceDialogue();
19    static void CloseDialogue();
20    static void SetInstantDisplay(bool _instantDisplay) { instantDisplay = _instantDisplay; }
21 };
22
```

11.118 DialogueUI.cpp File Reference

Class that stores the UI data for dialogue as well as this, it displays it correctly.

```
#include "DialogueUI.h"
#include "Cerberus/Core/Components/CAudioEmitterComponent.h"
```

11.118.1 Detailed Description

Class that stores the UI data for dialogue as well as this, it displays it correctly.

Author

Cathan Bertram

Date

May 2022

11.119 DialogueUI.h

```

1 #pragma once
2 #include <Cerberus/Core/CEntity.h>
3 #include <Cerberus/Core/Components/CSpriteComponent.h>
4 #include <Cerberus/Core/Components/CTextRenderComponent.h>
5
6 class CAudioEmitterComponent;
10 class DialogueUI : public CEntity
11 {
12 private:
13     CSpriteComponent* textBackground;
14     std::vector<CTextRenderComponent*> textRenderComponents;
15
16     CSpriteComponent* nameBackground;
17     CTextRenderComponent* nameTextRenderComponent;
18     CAudioEmitterComponent* audioEmitterComponent;
19
20     void UpdateTextComponentPosition(CTextRenderComponent* textComponent, int row);
21     float GetUIHeight();
22
23     float UIHeightPercent = 0.3f;
24     int maxCharactersInRow;
25     int maxRowCount;
26     int rowPadding = 4;
27     int rowHeight;
28     int charactersPerSecond = 50;
29     float timer = 0;
30     bool isUpdating = false;
31
32
33     std::string displayingText;
34     std::string reserveText;
35     std::string nameText;
36     void UpdateText();
37
38 public:
39     DialogueUI();
40     virtual ~DialogueUI();
41     virtual void Update(float deltaTime) override;
42
43     void SetText(const std::string& newText, bool instantDisplay);
44     void SetName(const std::string& newName);
45     void ClearText();
46     void Complete();
47     void CompletePage();
48     bool IsUpdating() { return isUpdating; }
49     bool IsComplete();
50     void Advance();
51     void ToggleDrawing(bool shouldDraw);
52     int GetReserveCharacterCount() { return reserveText.size(); }
53 };
54

```

11.120 IUsePickup.h

```

1 #pragma once
2 class IUsePickup
3 {
4 public:
5     virtual void UsePickup(const std::string& pickupToUse, float activeTime) = 0;
6 };

```


11.121 LevelTransporter.h

```

1 #pragma once
2 #include "Necrodoggiecon/Game/CInteractable.h"
3 #include "Necrodoggiecon/CWorld_Game.h"
4 class LevelTransporter : public CInteractable
5 {
6 public:
7     LevelTransporter();
8     void SetSlot(int SlotID) { Slot = SlotID; }
9
10    virtual void OnInteract();
11    int GetSlot() { return Slot; }
12 private:
13
14    int Slot;
15 };
16

```

11.122 NecrodoggieconPage.h

```

1 #pragma once
2 #include "Game/LevelTransporter.h"
3 class NecrodoggieconPage :
4     public LevelTransporter
5 {
6 public:
7     NecrodoggieconPage();
8     ~NecrodoggieconPage();
9     virtual void OnInteract() override;
10
11 protected:
12     void OnDialogueClose();
13 };
14

```

11.123 PlayerCharacter.h

```

1 #pragma once
2 #include <Necrodoggiecon\Game\CCharacter.h>
3 #include <Cerberus\Core\Environment\IInputable.h>
4 #include "Cerberus/Core/Components/CAudioEmitterComponent.h"
5 #include "Cerberus/Core/Utility/DebugOutput/Debug.h"
6 #include "IUsePickup.h"
7 #include "weapons.h"
8 #include <Necrodoggiecon/Weapons/Melee/Dagger.h>
9 #include <Necrodoggiecon/Weapons/Melee/Rapier.h>
10 #include <Necrodoggiecon/Weapons/Melee/Longsword.h>
11 #include <Necrodoggiecon/Weapons/Ranged/Crossbow.h>
12
13 class PlayerController;
14
15 class PlayerCharacter : public CCharacter, public IInputable, public IUsePickup
16 {
17 protected:
18     float walkSpeed = 300;
19     float walkDrag = 10;
20     float timeElapsed = 0;
21     float timeBetweenSteps = 0.35f;
22     float stepTimer;
23
24     void LookAt(Vector3 pos);
25
26     CAnimationSpriteComponent* spriteComponentBody = nullptr;
27     CAnimationSpriteComponent* spriteComponentLegs = nullptr;
28     CSpriteComponent* spriteComponentShadow = nullptr;
29     CSpriteComponent* spriteComponentShield = nullptr;
30     std::vector<PlayerController*> playersController = Engine::GetEntityOfType<PlayerController>();
31
32     Vector2 movementVec = { 0,0 };
33     XMFLOAT2 movementVel = { 0,0 };
34     XMFLOAT4 originalSpriteTint;
35     XMFLOAT4 originalLegTint;
36     const float walkAnimationSpeed = 1.3f;
37
38     float pickupTimer;
39     bool pickupActive;
40     float pickupActiveTime;
41

```

```

42     std::function<void()> pickupTimerCallback;
43     void InvisibilityCallback();
44     void PickupTimer(float deltaTime);
45
46     void ToggleVisibility(bool isVisible);
47     void ToggleShield(bool shield);
48     const float cameraMovementScalar = 100.0f;
49
50     bool hasShield = false;
51 public:
52     PlayerCharacter();
53
54     void PressedHorizontal(int dir, float deltaTime) override;
55     void PressedVertical(int dir, float deltaTime) override;
56     void PressedInteract() override;
57     void PressedDrop() override;
58     void Attack() override;
59     void PressedUse() override;
60
61     void UsePickup(const std::string& pickupToUse, float activeTime) override;
62     bool GetVisible() { return visible; }
63
64     virtual void Update(float deltaTime) override;
65     void EquipWeapon(Weapon* weapon);
66     void UpdateWeaponSprite();
67
68     void ApplyDamage(float damage);
69     void ApplyDamage(float damage, const std::string& onHitSound);
70
71     class CCameraComponent* camera = nullptr;
72
73 private:
74     void ResolveMovement(const float& deltaTime);
75     void AimAtMouse(const Vector3& mousePos);
76     void FootstepTimer(float deltaTime);
77 };
78

```

11.124 PlayerController.h

```

1 #pragma once
2 #include <Necrodoggiecon\Game\CPlayerController.h>
3 #include "PlayerCharacter.h"
4
5 class IInputable;
6
7 class PlayerController : public CPlayerController
8 {
9 public:
10     PlayerController();
11     virtual void Update(float deltaTime) override;
12
13     PlayerCharacter* charOne = nullptr;
14
15 protected:
16     virtual void HandleInput(float deltaTime) override;
17     int charIndex = 1;
18
19     IInputable* inputable = nullptr;
20
21     virtual void OnPossess() override;
22     virtual void OnUnpossess() override;
23
24     bool dialogueOpen = false;
25
26     void OnDialogueOpen() { dialogueOpen = true; }
27     void OnDialogueClose() { dialogueOpen = false; }
28 };
29

```

11.125 SoundManager.cpp File Reference

Static class used to handle the playing of audio within the game.

```

#include "SoundManager.h"
#include "Cerberus/Core/Engine.h"

```

11.125.1 Detailed Description

Static class used to handle the playing of audio within the game.

Author

Cathan Bertram

Date

May 2022

11.126 SoundManager.h

```

1 #pragma once
2 #include "Cerberus\Core\CEntity.h"
3 #include <Game/AudioEmitterEntity.h>
4
5 class SoundManager : public CEntity
6 {
7 public:
8     static void Initialise();
9     static void AddSound(const std::string& audioPath, const std::string& audioName, float audioRange);
10    static void AddSound(const std::string& audioPath, const std::string& audioName, float audioRange,
11        bool ambient);
12    static void PlaySound(const std::string& audioName, Vector3 position);
13    static void PlayMusic(const std::string& musicPath, CEntity* attachedEntity);
14 private:
15    static std::map<std::string, AudioEmitterEntity*> audioEmitterMap;
16    static AudioEmitterEntity* musicAudioEmitter;
17 };

```

11.127 TestUI.h

```

1 #pragma once
2 #include "Cerberus\Core\CEntity.h"
3 #include <array>
4
5 class TestUI : public CEntity
6 {
7     class CAnimationSpriteComponent* birb = nullptr;
8     class CTextRenderComponent* text1 = nullptr;
9     class CTextRenderComponent* text2 = nullptr;
10    class CTextRenderComponent* text3 = nullptr;
11    class CTextRenderComponent* textFPS = nullptr;
12    float timeElapsed = 0;
13    float textTimer = 0;
14    float fpsTimer = 0;
15    unsigned int framesTotal = 0;
16
17    const std::array<const char*, 6> texts =
18    {
19        "Wow",
20        "Amazing",
21        "Awesome",
22        "Nice One",
23        "uwu",
24        "Good Job",
25    };
26 public:
27     TestUI();
28     virtual void Update(float deltaTime) override;
29     virtual ~TestUI();
30 };
31

```

11.128 WeaponInterface.h File Reference

Interface class to implement the Weapons system using a Strategy Design Strategy.

```
#include "weapons.h"
#include "Cerberus/Core/CComponent.h"
#include "Cerberus\Core\Engine.h"
```

Classes

- class [WeaponInterface](#)
Weapon Interface class used to switch weapons being used through the Strategy Design Pattern.

11.128.1 Detailed Description

Interface class to implement the Weapons system using a Strategy Design Strategy.

Author

Ben Brown

Date

May 2022

11.129 WeaponInterface.h

[Go to the documentation of this file.](#)

```
1  /*****
8  #pragma once
9  #include "weapons.h"
10 #include "Cerberus/Core/CComponent.h"
11
12 #include "Cerberus\Core\Engine.h"
13
17 class WeaponInterface : public CComponent
18 {
19 public:
20     WeaponInterface();
21     ~WeaponInterface();
22
23     virtual bool OnFire(Vector3 actorPos, Vector3 attackDir);
24     virtual void Update(float deltaTime) override;
25     virtual void Draw(ID3D11DeviceContext* context, const XMFLOAT4X4& parentMat, ConstantBuffer cb,
26         ID3D11Buffer* constantBuffer) override;
27
28     void SetWeapon(Weapon* weapon);
29     Weapon* GetCurrentWeapon() { return currentWeapon; };
30
31     void SetUserType(USERTYPE userType);
32     USERTYPE GetUserType() { return currentWeapon->GetUserType(); };
33 private:
34     Weapon* currentWeapon = nullptr;
35
36     USERTYPE userType = USERTYPE::AI;
37 };
```

11.130 WeaponPickup.h File Reference

A class that inherits from [CInteractable](#) which allows for weapons to be spawned within the world and picked up by the player.

```
#include "Necrodoggiecon/Game/CInteractable.h"
#include "Necrodoggiecon/Game/weapons.h"
#include "Cerberus/Core/Utility/DebugOutput/Debug.h"
#include "Necrodoggiecon/Game/PlayerCharacter.h"
#include "Cerberus/Core/Utility/IO.h"
#include "Cerberus/Core/Components/CAudioEmitterComponent.h"
#include "Game/SoundManager.h"
```

Classes

- class [WeaponPickup< T >](#)

11.130.1 Detailed Description

A class that inherits from [CInteractable](#) which allows for weapons to be spawned within the world and picked up by the player.

Author

Luke Whiting

Date

May 2022

11.131 WeaponPickup.h

[Go to the documentation of this file.](#)

```
1  /*****
9  #pragma once
10 #include "Necrodoggiecon/Game/CInteractable.h"
11 #include "Necrodoggiecon/Game/weapons.h"
12 #include "Cerberus/Core/Utility/DebugOutput/Debug.h"
13 #include "Necrodoggiecon/Game/PlayerCharacter.h"
14 #include "Cerberus/Core/Utility/IO.h"
15 #include "Cerberus/Core/Components/CAudioEmitterComponent.h"
16 #include "Game/SoundManager.h"
17 template<typename T>
18 class WeaponPickup : public CInteractable
19 {
20 public:
21     WeaponPickup();
22     virtual ~WeaponPickup();
23
24     virtual void OnInteract() override;
25
26     void SetWeapon(T* weapon);
27
28 private:
29
30     void UpdateWeaponSprite(Weapon* weapon);
31
32     Weapon* pickup = nullptr;
```

```

33 };
34
35 template<typename T>
36 inline WeaponPickup<T>::WeaponPickup()
37 {
38     T* weapon = new T();
39     Weapon* baseWeapon = dynamic_cast<Weapon*>(weapon);
40     if (baseWeapon != nullptr)
41     {
42         pickup = baseWeapon;
43         UpdateWeaponSprite(weapon);
44     }
45     else
46     {
47         Debug::LogError("Tried to create a entity with invalid type: %s", typeid(*weapon).name());
48         delete weapon;
49         return;
50     }
51 };
52
53 template<typename T>
54 inline WeaponPickup<T>::~WeaponPickup()
55 {
56     delete pickup;
57     pickup = nullptr;
58 }
59
60 template<typename T>
61 inline void WeaponPickup<T>::OnInteract()
62 {
63     PlayerCharacter* player = dynamic_cast<PlayerCharacter*>(this->GetLastCollidedObject()->GetParent());
64     if (player != nullptr)
65     {
66         if (this->pickup != nullptr)
67         {
68             Weapon* pickupDupe = this->pickup;
69             Weapon* playerDupe = player->GetWeapon();
70
71             player->EquipWeapon(pickupDupe);
72             this->pickup = playerDupe;
73             SoundManager::PlaySound("ItemPickup", GetPosition());
74             UpdateWeaponSprite(this->pickup);
75         }
76         else
77         {
78             Debug::LogError("Tried to interact with a weapon pickup that doesnt have one set!.");
79             return;
80         }
81     }
82     else
83     {
84         Debug::LogError("Tried to interact with a weapon when not the player character!.");
85         return;
86     }
87 }
88
89 template<typename T>
90 inline void WeaponPickup<T>::SetWeapon(T* weapon)
91 {
92     Weapon* baseWeapon = dynamic_cast<Weapon*>(weapon);
93     if (baseWeapon != nullptr)
94     {
95         pickup = weapon;
96         UpdateWeaponSprite(baseWeapon);
97     }
98     else
99     {
100         Debug::LogError("Tried to set weapon on pickup to a type that isnt a weapon. Type: %s",
101             typeid(*weapon).name());
102         return;
103     }
104 }
105
106 template<typename T>
107 inline void WeaponPickup<T>::UpdateWeaponSprite(Weapon* weapon)
108 {
109     std::string ext = IO::FindExtension(weapon->GetIconPath());
110     CSpriteComponent* sprite = this->GetSprite();
111     if (ext == ".dds")
112     {
113         sprite->LoadTexture(weapon->GetIconPath());
114         sprite->SetTextureOffset(weapon->GetTextureOffset());
115         sprite->SetRenderRect(weapon->GetRenderRect());
116         sprite->SetScale(weapon->GetScale());
117     }
118     else

```

```

133     {
134         sprite->LoadTextureWIC(weapon->GetIconPath());
135         sprite->SetTextureOffset(weapon->GetTextureOffset());
136         sprite->SetRenderRect(weapon->GetRenderRect());
137         sprite->SetScale(weapon->GetScale());
138     }
139 }

```

11.132 weapons.h File Reference

Base [Weapon](#) class for the weapons in the game, this will be inherited by the custom classes of the weapons.

```

#include <string>
#include <fstream>
#include "Necrodoggiecon/Projectile.h"
#include "Cerberus/Core/CComponent.h"
#include "Cerberus/Core/CEntity.h"
#include "Cerberus\Core\Engine.h"
#include "Cerberus/Core/Utility/DebugOutput/Debug.h"
#include "Cerberus\Core\Utility\Vector3.h"
#include "Cerberus\Dependencies\NlohmannJson\json.hpp"

```

Classes

- class [Weapon](#)
Base [Weapon](#) class inherited by all weapons.

Macros

- #define **rangeScale** 64.0f

Typedefs

- using **json** = nlohmann::json

Enumerations

- enum class **USERTYPE** { **PLAYER** , **AI** }

11.132.1 Detailed Description

Base [Weapon](#) class for the weapons in the game, this will be inherited by the custom classes of the weapons.

Author

Ben Brown & Flynn Brooks

Date

May 2022

11.133 weapons.h

[Go to the documentation of this file.](#)

```
1  /*****
8  #pragma once
9  #include <string>
10 #include <fstream>
11
12 #include "Necrodoggiecon/Projectile.h"
13 #include "Cerberus/Core/CComponent.h"
14 #include "Cerberus/Core/CEntity.h"
15 #include "Cerberus/Core/Engine.h"
16 #include "Cerberus/Core/Utility/DebugOutput/Debug.h"
17 #include "Cerberus/Core/Utility/Vector3.h"
18 #include "Cerberus/Dependencies/NlohmannJson/json.hpp"
19
20 #define rangeScale 64.0f
21
22 using json = nlohmann::json;
23
24 enum class USERTYPE
25 {
26     PLAYER,
27     AI,
28 };
29
30 class Weapon : public CComponent
31 {
32 public:
33     Weapon(std::string weapon = "Dagger");
34
35     virtual bool OnFire(Vector3 actorPos, Vector3 attackDir);
36     void SetWeapon(int ID);
37     void SetWeapon(std::string ID);
38
39     std::string IDToName(int ID);
40     int NameToID(std::string Name);
41
42     virtual void Update(float deltaTime) override;
43     virtual void Draw(ID3D11DeviceContext* context, const XMFLOAT4X4& parentMat, ConstantBuffer cb,
44         ID3D11Buffer* constantBuffer) override;
45
46     void SetUserType(USERTYPE userType) { this->userType = userType; };
47
48     std::string GetType() { return type; };
49     std::string GetProjectileIcon() { return projectileIconPath; };
50     float GetDamage() { return damage; };
51     float GetRange() { return range; };
52     float GetAttack_Speed() { return attack_speed; };
53     float GetMaxAmmo() { return maxAmmo; };
54     void SetMaxAmmo(float amount) { maxAmmo = amount; };
55     float GetAmmo() { return ammo; };
56     void SetAmmo(float amount) { ammo = amount; };
57     bool GetUnique() { return unique; };
58     bool GetCanFire() { return canFire; };
59     void SetCanFire(bool canFire) { this->canFire = canFire; };
60     void SetTextureOffset(XMFLOAT2 offset) { textureOffset = offset; };
61     XMFLOAT2 GetTextureOffset() { return textureOffset; };
62     void SetRenderRect(XMUINT2 rect) { renderRect = rect; };
63     XMUINT2 GetRenderRect() { return renderRect; };
64     void SetScale(XMFLOAT3 setScale) { scale = setScale; };
65     XMFLOAT3 GetScale() { return scale; };
66     USERTYPE GetUserType() { return userType; };
67     std::string GetName() { return name; };
68     std::string GetIconPath() { return iconPath; };
69     std::string GetHitSound() { return hitSound; };
70     std::string GetAttackSound() { return attackSound; };
71
72     void StartCooldown() { cooldown = attack_speed; };
73
74 private:
75     void CoolDown(float attack_cooldown);
76
77     std::string iconPath;
78     std::string projectileIconPath;
79     std::string type;
80     std::string name;
81     std::string hitSound;
82     std::string attackSound;
83     float damage;
84     float range;
85     float attack_speed;
86     float ammo;
```



```
91     float maxAmmo;
92     bool unique;
93     bool canFire = true;
94     float cooldown;
95
96     XMFLOAT2 textureOffset = XMFLOAT2(0.0, 0.0);
97     XMUINT2 renderRect = XMUINT2(64, 64);
98     XMFLOAT3 scale = XMFLOAT3(1.0, 1.0, 1.0);
99
100     USERTYPE userType;
101
102 protected:
103     std::string pickupType;
104
105 };
106
```

11.134 HomingProjectile.cpp File Reference

All the functions needed for Homing [Projectile](#).

```
#include "HomingProjectile.h"
#include "Necrodoggiecon\Game\AI\CAIController.h"
#include "Necrodoggiecon/Game/PlayerCharacter.h"
```

11.134.1 Detailed Description

All the functions needed for Homing [Projectile](#).

Author

Flynn Brooks

Date

May 2022

11.135 HomingProjectile.h File Reference

Header containing all the functions and variables needed for Homing [Projectile](#).

```
#include <Necrodoggiecon/Projectile.h>
#include <Necrodoggiecon\Game\CCharacter.h>
```

Classes

- class [HomingProjectile](#)

11.135.1 Detailed Description

Header containing all the functions and variables needed for Homing [Projectile](#).

Author

Flynn Brooks

Date

May 2022

11.136 HomingProjectile.h

[Go to the documentation of this file.](#)

```
1  /*****/
9  #pragma once
10 #include <Necrodoggiecon/Projectile.h>
11 #include <Necrodoggiecon\Game\CCharacter.h>
12
13 class HomingProjectile : public Projectile
14 {
15 public:
16     HomingProjectile();
17     ~HomingProjectile();
18
19     virtual void Update(float deltaTime);
20 private:
21     CAIController* GetClosestEnemy(Vector3 actorPos, float ranged);
22     CCharacter* GetClosestPlayer(Vector3 actorPos, float ranged);
23 };
24
```

11.137 LevelCompleteMenu.cpp File Reference

cpp for setting up the level complete screen

```
#include "LevelCompleteMenu.h"
#include "Cerberus/Core/UI/CWidget_Button.h"
#include "Cerberus/Core/UI/CWidget_Image.h"
#include "Cerberus/Core/Components/CTextRenderComponent.h"
#include "Cerberus/Core/Utility/CWorldManager.h"
#include "CWorld_Game.h"
#include "Cerberus/Core/UI/CWidget_Text.h"
#include "Cerberus/Core/Utility/CUIManager.h"
#include "Game/SoundManager.h"
#include "Necrodoggiecon/CWorld_Menu.h"
```

11.137.1 Detailed Description

cpp for setting up the level complete screen

Author

Jack B

Date

May 2022

11.138 LevelCompleteMenu.h File Reference

Header for the level complete screen.

```
#include "Cerberus/Core/UI/CWidget_Canvas.h"
```

Classes

- class [LevelCompleteMenu](#)

11.138.1 Detailed Description

Header for the level complete screen.

Author

Jack B

Date

May 2022

11.139 LevelCompleteMenu.h

[Go to the documentation of this file.](#)

```
1 /*****/
2 #pragma once
3 #include "Cerberus/Core/UI/CWidget_Canvas.h"
4
5 class LevelCompleteMenu : public CWidget_Canvas
6 {
7     virtual void InitialiseCanvas() override;
8
9 public:
10     LevelCompleteMenu();
11     void QuitToMenu();
12     void QuitToDesktop();
13     void NextLevel();
14 };
15
```

11.140 LevelSelectMenu.cpp File Reference

The cpp for the level select menu.

```
#include "LevelSelectMenu.h"
#include "Cerberus/Core/UI/CWidget_Button.h"
#include "Cerberus/Core/UI/CWidget_Image.h"
#include "Cerberus/Core/Components/CTextRenderComponent.h"
#include "Cerberus/Core/Utility/CUIManager.h"
#include "Cerberus/Core/UI/CWidget_Text.h"
#include "Cerberus/Core/Utility/CWorldManager.h"
#include "CWorld_Game.h"
#include "Game/SoundManager.h"
```

11.140.1 Detailed Description

The cpp for the level select menu.

Author

Jack B

Date

May 2022

11.141 LevelSelectMenu.h File Reference

Header for the level select menu.

```
#include "Cerberus/Core/UI/CWidget_Canvas.h"
```

Classes

- class [LevelSelectMenu](#)

11.141.1 Detailed Description

Header for the level select menu.

Author

Jack B

Date

May 2022

11.142 LevelSelectMenu.h

[Go to the documentation of this file.](#)

```
1  /*****
2  #pragma once
3  #include "Cerberus/Core/UI/CWidget_Canvas.h"
4
5  11 class LevelSelectMenu : public CWidget_Canvas
6  12 {
7  13     virtual void InitialiseCanvas() override;
8  14
9  15     int SelectedLevel = 0;
10 16
11 17     CWidget_Button* LVL0;
12 18     CWidget_Button* LVL1;
13 19     CWidget_Button* LVL2;
14 20     CWidget_Button* LVL3;
15 21     CWidget_Button* LVL4;
16 22     CWidget_Button* LVL5;
17 23     CWidget_Button* LVL6;
18 24     CWidget_Button* LVL7;
19 25
20 26 public:
21 27     LevelSelectMenu();
22 28     void CloseMenu();
23 29
24 30     void OpenLevelTutorial();
25 31     void OpenLevel1();
26 32     void OpenLevel2();
27 33     void OpenLevel3();
28 34     void OpenLevel4();
29 35     void OpenLevel5();
30 36     void OpenLevel6();
31 37     void OpenLevel7();
32 38
33 39     void UpdateButtonPositions();
34 40
35 41     void PlayLevel();
36 42
37 43 };
38 44
```

11.143 MainMenu.cpp File Reference

The cpp for the main menu.

```
#include "MainMenu.h"
#include "Cerberus/Core/UI/CWidget_Button.h"
#include "Cerberus/Core/UI/CWidget_Image.h"
#include "Cerberus/Core/UI/CWidget_Text.h"
#include "Cerberus/Core/Utility/CWorldManager.h"
#include "CWorld_Game.h"
#include "Cerberus/Core/Utility/CUIManager.h"
#include "SettingsMenu.h"
#include "LevelSelectMenu.h"
#include "Game/SoundManager.h"
```

11.143.1 Detailed Description

The cpp for the main menu.

Author

Jack B

Date

May 2022

11.144 MainMenu.h File Reference

Header for the main menu.

```
#include "Cerberus/Core/UI/CWidget_Canvas.h"
```

Classes

- class [MainMenu](#)

11.144.1 Detailed Description

Header for the main menu.

Author

Jack B

Date

May 2022

11.145 MainMenu.h

[Go to the documentation of this file.](#)

```
1  /*****  
8  #pragma once  
9  #include "Cerberus/Core/UI/CWidget_Canvas.h"  
10 class MainMenu :  
11     public CWidget_Canvas  
12 {  
13  
14  
15     virtual void InitialiseCanvas() override;  
16  
17  
18 public:  
19     MainMenu();  
20  
21     void QuitToDesktop();  
22  
23     void OpenLevelSelect();  
24     void OpenSettingsMenu();  
25  
26 };  
27
```

11.146 PauseMenu.cpp File Reference

The cpp for the pause menu.

```
#include "PauseMenu.h"
#include "Cerberus/Core/UI/CWidget_Button.h"
#include "Cerberus/Core/UI/CWidget_Image.h"
#include "Cerberus/Core/Components/CTextRenderComponent.h"
#include "Cerberus/Core/Utility/CWorldManager.h"
#include "CWorld_Game.h"
#include "Cerberus/Core/UI/CWidget_Text.h"
#include "Cerberus/Core/Utility/CUIManager.h"
#include "SettingsMenu.h"
#include "LevelCompleteMenu.h"
#include "Game/SoundManager.h"
#include "Necrodoggiecon/CWorld_Menu.h"
```

11.146.1 Detailed Description

The cpp for the pause menu.

Author

Jack B

Date

May 2022

11.147 PauseMenu.h File Reference

Header for the pause menu.

```
#include "Cerberus/Core/UI/CWidget_Canvas.h"
```

Classes

- class [PauseMenu](#)

11.147.1 Detailed Description

Header for the pause menu.

Author

Jack B

Date

May 2022

11.148 PauseMenu.h

[Go to the documentation of this file.](#)

```
1 /*****
2  *****/
3 #pragma once
4 #include "Cerberus/Core/UI/CWidget_Canvas.h"
5
6 class PauseMenu : public CWidget_Canvas
7 {
8     virtual void InitialiseCanvas() override;
9
10    bool isPaused = false;
11    bool gameEnded = false;
12
13 private:
14
15    bool pausePressedDown = false;
16
17 public:
18    PauseMenu();
19
20    void PauseGame();
21    void ResumeGame();
22    void QuitToMenu();
23    void QuitToDesktop();
24    void OpenSettingsMenu();
25
26    virtual void Update(float deltaTime) override;
27 };
28
29
```

11.149 Projectile.cpp File Reference

All the functions needed for the [Projectile](#).

```
#include "Projectile.h"
#include "Necrodoggiecon\Game\AI\CAIController.h"
#include <Necrodoggiecon\Game\PlayerCharacter.h>
#include <Cerberus/Core/Components/CAudioEmitterComponent.h>
```

11.149.1 Detailed Description

All the functions needed for the [Projectile](#).

Author

Flynn Brooks

Date

May 2022

11.150 Projectile.h File Reference

Header containing all the functions and variables needed for the [Projectile](#).

```
#include <Cerberus\Core\Components\CAnimationSpriteComponent.h>
#include <Cerberus\Core\CEntity.h>
```


Classes

- class [Projectile](#)
Projectile class for the [Projectile](#).

Enumerations

- enum class **USERTYPE2** { **PLAYER** , **AI** }

11.150.1 Detailed Description

Header containing all the functions and variables needed for the [Projectile](#).

Author

Flynn Brooks

Date

May 2022

11.151 Projectile.h

[Go to the documentation of this file.](#)

```
1  /*****/
9  #pragma once
10 #include <Cerberus\Core\Components\CAnimationSpriteComponent.h>
11 #include <Cerberus\Core\CEntity.h>
12
13 class CAudioEmitterComponent;
14 class CAIController;
15 class PlayerCharacter;
16
17 enum class USERTYPE2
18 {
19     PLAYER,
20     AI,
21 };
22
26 class Projectile : public CEntity
27 {
28 public:
29
30     Projectile();
31     ~Projectile();
32
33     void StartUp(Vector3 dir, Vector3 pos, float damage, float speed, float lifetime, int type, const
std::string &projectile_name, const std::string& hitAudioPath);
34     void DidItHit();
35     virtual void Update(float deltaTime) override;
36
37     void SetLifetime(float life) { Lifetime = life; }
38     float GetLifetime() { return Lifetime; };
39     Vector3 GetPosition() { return Position; };
40     void SetPosition(Vector3 newPosition) { Position = newPosition; };
41     Vector3 GetDirection() { return Direction; };
42     float GetSpeed() { return Speed; };
43
44     USERTYPE2 GetUserType() { return userType; };
45     class CSpriteComponent* ProjectileSprite = nullptr;
46
47 private:
48     float Damage;
49
49     float Speed;
```

```

51     float Lifetime;
52     float damage;
53     Vector3 velocity = { 0.0f, 0.0f, 0.0f };
54     Vector3 acceleration = { 0.0f, 0.0f, 0.0f };
55     Vector3 Direction;
56     Vector3 Position;
57     Vector3 initialPosition;
58     std::string Projectile_Name;
59     std::string onHitAudioPath;
60     bool hasHit = false;
61
62     CAIController* GetClosestEnemy(Vector3 actorPos);
63     PlayerCharacter* GetClosestPlayer(Vector3 actorPos);
64     CAIController* GetClosestEnemy(Vector3 actorPos, float ranged);
65     USERTYPE2 userType;
66 };
67

```

11.152 SettingsMenu.cpp File Reference

The cpp for the settings menu.

```

#include "SettingsMenu.h"
#include "Cerberus/Core/UI/CWidget_Button.h"
#include "Cerberus/Core/UI/CWidget_Image.h"
#include "Cerberus/Core/Components/CTextRenderComponent.h"
#include "Cerberus/Core/Utility/CUIManager.h"
#include "Cerberus/Core/UI/CWidget_Text.h"
#include "Game/SoundManager.h"
#include "Cerberus\Core\Utility\Audio\AudioController.h"

```

11.152.1 Detailed Description

The cpp for the settings menu.

Author

Jack B

Date

May 2022

11.153 SettingsMenu.h File Reference

Header for the settings menu.

```

#include "Cerberus/Core/UI/CWidget_Canvas.h"

```

Classes

- class [SettingsMenu](#)

11.153.1 Detailed Description

Header for the settings menu.

Author

Jack B

Date

May 2022

11.154 SettingsMenu.h

[Go to the documentation of this file.](#)

```
1  /*****
2  8  #pragma once
3  9  #include "Cerberus/Core/UI/CWidget_Canvas.h"
4  10
5  11 class SettingsMenu : public CWidget_Canvas
6  12 {
7  13     virtual void InitialiseCanvas() override;
8  14
9  15 public:
10 16     SettingsMenu();
11 17     void CloseSettings();
12 18
13 19     virtual void Update(float deltaTime) override;
14 20
15 21 private:
16 22
17 23     CWidget_Text* CreateVolumeUI(Vector2 pos, const std::string& title, const int& volume,
18 24         std::function<void()> volumeUp, std::function<void()> volumeDown);
19 25
20 26     void MasterVolumeUp();
21 27     void MasterVolumeDown();
22 28
23 29     CWidget_Text* masterVolumeText;
24 30
25 31     int masterVolume = 100;
26 32 };
```

11.155 Dagger.h File Reference

Sub-Class for the [Dagger](#) weapon.

```
#include <Necrodoggiecon/Weapons/MeleeWeapon.h>
```

Classes

- class [Dagger](#)

11.155.1 Detailed Description

Sub-Class for the [Dagger](#) weapon.

Author

Ben Brown

Date

May 2022

11.156 Dagger.h

[Go to the documentation of this file.](#)

```
1 /*****  
9 #pragma once  
10 #include <Necrodoggiecon/Weapons/MeleeWeapon.h>  
11  
12 class Dagger : public MeleeWeapon  
13 {  
14 public:  
15     Dagger();  
16     ~Dagger();  
17  
18 private:  
19  
20 };  
21
```

11.157 Longsword.h File Reference

Sub-Class for the [Longsword](#) weapon.

```
#include <Necrodoggiecon/Weapons/MeleeWeapon.h>
```

Classes

- class [Longsword](#)

11.157.1 Detailed Description

Sub-Class for the [Longsword](#) weapon.

This will include all unique logic for the weapon (AOE Slashing)

Author

Ben Brown

Date

May 2022

11.158 Longsword.h

[Go to the documentation of this file.](#)

```
1 /*****
9 #pragma once
10 #include <Necrodoggiecon/Weapons/MeleeWeapon.h>
11
12 class Longsword : public MeleeWeapon
13 {
14 public:
15     Longsword();
16     ~Longsword();
17
18     virtual bool OnFire(Vector3 actorPos, Vector3 attackDir);
19 private:
20     std::vector<CEntity*> GetPlayersInReach(Vector3 actorPos, Vector3 damagePos);
21 };
22
23
```

11.159 Rapier.h File Reference

Sub-Class for the [Rapier](#) weapon.

```
#include <Necrodoggiecon/Weapons/MeleeWeapon.h>
```

Classes

- class [Rapier](#)

11.159.1 Detailed Description

Sub-Class for the [Rapier](#) weapon.

This holds all unique logic for the weapon

Author

Ben Brown

Date

May 2022

11.160 Rapier.h

[Go to the documentation of this file.](#)

```
1 /*****
8 #pragma once
9 #include <Necrodoggiecon/Weapons/MeleeWeapon.h>
10
11 class Rapier : public MeleeWeapon
12 {
13 public:
14     Rapier();
15     ~Rapier();
16
17 private:
18
19 };
20
```

11.161 MeleeWeapon.cpp File Reference

Base Melee [Weapon](#) class that all Sub-Classes of melee weapons inherit from.

```
#include "MeleeWeapon.h"
#include "Necrodoggiecon\Game\PlayerCharacter.h"
#include "Necrodoggiecon\Game\AI\CAIController.h"
```

11.161.1 Detailed Description

Base Melee [Weapon](#) class that all Sub-Classes of melee weapons inherit from.

Author

Ben Brown

Date

May 2022

11.162 MeleeWeapon.h

```
1 #pragma once
2 #include <Necrodoggiecon\Game\weapons.h>
3 #include <Necrodoggiecon\Game\CCharacter.h>
4
5 class MeleeWeapon : public Weapon
6 {
7 public:
8     MeleeWeapon();
9     ~MeleeWeapon();
10
11     virtual bool OnFire(Vector3 actorPos, Vector3 attackDir);
12 private:
13     CCharacter* GetClosestEnemy(Vector3 actorPos, Vector3 damagePos);
14     CCharacter* GetClosestPlayer(Vector3 actorPos, Vector3 damagePos);
15
16     void HandleMelee(Vector3 actorPos, Vector3 normAttackDir);
17 };
18
```

11.163 Pickup.cpp File Reference

Class to handle scroll pickups.

```
#include "Pickup.h"
#include "Necrodoggiecon\Game\PlayerCharacter.h"
#include "Necrodoggiecon\Game\AI\CAIController.h"
```

11.163.1 Detailed Description

Class to handle scroll pickups.

Author

Cathan Bertram

Date

May 2022

11.164 Pickup.h

```
1 #pragma once
2 #include <Necrodoggiecon\Game\weapons.h>
3
4 class Pickup : public Weapon
5 {
6 public:
7     Pickup();
8     ~Pickup();
9     void Update(float deltaTime) override;
10    virtual bool OnFire(Vector3 actorPos, Vector3 attackDir);
11 private:
12    CEntity* GetClosestEnemy(Vector3 actorPos, Vector3 damagePos);
13    CEntity* GetClosestPlayer(Vector3 actorPos, Vector3 damagePos);
14
15    void HandlePickup();
16 };
17
```

11.165 InvisibilityScroll.h

```
1 #pragma once
2 #include "Necrodoggiecon/Weapons/Pickup.h"
3 class InvisibilityScroll :
4     public Pickup
5 {
6 public:
7     InvisibilityScroll();
8     ~InvisibilityScroll();
9 };
10
```

11.166 ShieldScroll.h

```
1 #pragma once
2 #include "Necrodoggiecon/Weapons/Pickup.h"
3 class ShieldScroll :
4     public Pickup
5 {
6 public:
7     ShieldScroll();
8     ~ShieldScroll();
9 };
10
```

11.167 Crossbow.cpp File Reference

All the functions needed for [Crossbow](#).

```
#include "Crossbow.h"
```

11.167.1 Detailed Description

All the functions needed for [Crossbow](#).

Author

Flynn Brooks

Date

May 2022

11.168 Crossbow.h File Reference

Header containing all the functions and variables needed for [Crossbow](#).

```
#include <Necrodoggiecon/Weapons/RangeWeapon.h>
```

Classes

- class [Crossbow](#)

11.168.1 Detailed Description

Header containing all the functions and variables needed for [Crossbow](#).

Author

Flynn Brooks

Date

May 2022

11.169 Crossbow.h

[Go to the documentation of this file.](#)

```
1  /*****  
9  #pragma once  
10 #include <Necrodoggiecon/Weapons/RangeWeapon.h>  
11  
12 class Crossbow : public RangeWeapon  
13 {  
14 public:  
15     Crossbow();  
16     ~Crossbow();  
17  
18     virtual void Update(float deltaTime);  
19 };  
20
```


11.170 Fireball.cpp File Reference

All the functions needed for fireball.

```
#include "Fireball.h"
```

11.170.1 Detailed Description

All the functions needed for fireball.

Author

Flynn Brooks

Date

May 2022

11.171 Fireball.h File Reference

Header containing all the functions and variables needed for FireBall.

```
#include <Necrodoggiecon/Weapons/RangeWeapon.h>
```

Classes

- class [Fireball](#)

11.171.1 Detailed Description

Header containing all the functions and variables needed for FireBall.

Author

Flynn Brooks

Date

May 2022

11.172 Fireball.h

[Go to the documentation of this file.](#)

```
1 /*****  
9 #pragma once  
10 #include <Necrodoggiecon/Weapons/RangeWeapon.h>  
11 class Fireball : public RangeWeapon  
12 {  
13 public:  
14     Fireball();  
15     ~Fireball();  
16 private:  
17  
18 };  
19
```

11.173 MagicMissile.cpp File Reference

All the functions needed for Magic Missile.

```
#include "MagicMissile.h"
```

11.173.1 Detailed Description

All the functions needed for Magic Missile.

Author

Flynn Brooks

Date

May 2022

11.174 MagicMissile.h File Reference

Header containing all the functions and variables needed for the Magic Missile.

```
#include <Necrodoggiecon/Weapons/RangeWeapon.h>  
#include <Necrodoggiecon/HomingProjectile.h>
```

Classes

- class [MagicMissile](#)

11.174.1 Detailed Description

Header containing all the functions and variables needed for the Magic Missile.

Author

Flynn Brooks

Date

May 2022

11.175 MagicMissile.h

[Go to the documentation of this file.](#)

```
1 /*****
9 #pragma once
10 #include <Necrodoggiecon/Weapons/RangeWeapon.h>
11 #include <Necrodoggiecon/HomingProjectile.h>
12
13 class MagicMissile : public RangeWeapon
14 {
15 public:
16     MagicMissile();
17     ~MagicMissile();
18
19     virtual bool OnFire(Vector3 actorPos, Vector3 attackDir);
20 private:
21
22 };
23
```

11.176 RangeWeapon.h

```
1 /*****
9 #pragma once
10 #include <Necrodoggiecon/Game/weapons.h>
11 class RangeWeapon : public Weapon
12 {
13 public:
14     RangeWeapon();
15     ~RangeWeapon();
16
17     virtual bool OnFire(Vector3 actorPos, Vector3 attackDir);
18
19     void SetProjectileSpeed(float speed) { projectileSpeed = speed; };
20     float GetProjectileSpeed() { return projectileSpeed; };
21 private:
22     void HandleRanged(Vector3 actorPos, Vector3 attackDir);
23     float projectileSpeed = 4;
24 };
25
```

11.177 weaponUI.cpp File Reference

This is the CPP for the weapon UI and the timer.

```
#include "weaponUI.h"
#include <sstream>
#include "Cerberus/Core/Utility/Math/Math.h"
#include "Cerberus\Core\Components\CTextRenderComponent.h"
#include "Cerberus\Core\Components\CSpriteComponent.h"
#include "Cerberus\Core\Structs\CCamera.h"
```

11.177.1 Detailed Description

This is the CPP for the weapon UI and the timer.

Author

Jack B

Date

May 2022

11.178 weaponUI.h File Reference

Header file for the weapon UI.

```
#include "Cerberus\Core\CEntity.h"
```

Classes

- class [weaponUI](#)

11.178.1 Detailed Description

Header file for the weapon UI.

Author

Jack B

Date

May 2022

11.179 weaponUI.h

[Go to the documentation of this file.](#)

```
1  /*****
8  #pragma once
9  #include "Cerberus\Core\CEntity.h"
10
11 class weaponUI : public CEntity
12 {
13     class CSpriteComponent* spriteBack = nullptr;
14     class CSpriteComponent* ammoBack = nullptr;
15     class CSpriteComponent* weaponSprite = nullptr;
16     class CTextRenderComponent* textWeaponName = nullptr;
17     class CTextRenderComponent* textAmmoDisplay = nullptr;
18     class CTextRenderComponent* textTimer = nullptr;
19
20     float seconds = 0;
21     int minutes = 0;
22
23 public:
24     weaponUI();
25     virtual void updateUI(std::string WeaponName, int currentAmmo, int maxAmmo, std::string spritePath);
26     virtual void Update(float deltaTime) override;
27     virtual ~weaponUI();
28 };
```

Index

- [_Material](#), [31](#)
- [AddAudio](#)
 - [AssetManager](#), [34](#)
- [AddCamera](#)
 - [CameraManager](#), [53](#)
- [AddChild](#)
 - [CWidget](#), [111](#)
- [AddEmitter](#)
 - [AudioController](#), [38](#)
- [AddListener](#)
 - [AudioController](#), [39](#)
 - [EventSystem](#), [141](#)
- [AddMesh](#)
 - [AssetManager](#), [34](#)
- [AddSound](#)
 - [SoundManager](#), [184](#)
- [Advance](#)
 - [DialogueUI](#), [135](#)
- [AdvanceDialogue](#)
 - [DialogueHandler](#), [133](#)
- [AlarmEnemy](#), [31](#)
 - [ChasePlayer](#), [32](#)
 - [OnDeath](#), [32](#)
 - [OnHit](#), [32](#)
 - [Update](#), [33](#)
- [AlarmEnemy.cpp](#), [259](#)
- [AlarmEnemy.h](#), [260](#)
- [ApplyDamage](#)
 - [CAIController](#), [48](#)
 - [CCharacter](#), [64](#)
 - [PlayerCharacter](#), [172](#)
- [AssetManager](#), [33](#)
 - [AddAudio](#), [34](#)
 - [AddMesh](#), [34](#)
 - [GetAudio](#), [34](#)
 - [GetDefaultMesh](#), [35](#)
 - [GetMesh](#), [35](#)
 - [GetTexture](#), [35](#)
 - [GetTextureWIC](#), [36](#)
 - [RemoveAudio](#), [36](#)
- [AssetManager.h](#), [229](#), [230](#)
- [Attack](#)
 - [PlayerCharacter](#), [172](#)
- [AttackEnter](#)
 - [CAIController](#), [49](#)
 - [DogEnemy](#), [137](#)
- [AttackPlayer](#)
 - [CAIController](#), [49](#)
 - [DogEnemy](#), [138](#)
- [GruntEnemy](#), [143](#)
- [AttackState](#), [36](#)
 - [Enter](#), [37](#)
 - [Exit](#), [37](#)
 - [Update](#), [37](#)
- [AudioController](#), [38](#)
 - [AddEmitter](#), [38](#)
 - [AddListener](#), [39](#)
 - [DestroyAudio](#), [39](#)
 - [GetAllEmittersWithinRange](#), [39](#)
 - [LoadAudio](#), [40](#)
 - [PlayAudio](#), [40](#)
 - [RemoveEmitter](#), [41](#)
 - [StopAudio](#), [41](#)
- [AudioController.h](#), [230](#), [231](#)
- [AudioEmitterEntity](#), [42](#)
 - [Load](#), [43](#)
 - [PlayAudio](#), [43](#), [44](#)
 - [SetAudio](#), [44](#)
 - [SetRange](#), [45](#)
 - [Update](#), [45](#)
- [AudioEmitterEntity.cpp](#), [268](#)
- [AudioEmitterEntity.h](#), [269](#)
- [Bind_HoverEnd](#)
 - [CWidget_Button](#), [113](#)
- [Bind_HoverStart](#)
 - [CWidget_Button](#), [114](#)
- [Bind_OnButtonPressed](#)
 - [CWidget_Button](#), [114](#)
- [Bind_OnButtonReleased](#)
 - [CWidget_Button](#), [114](#)
- [CAIController](#), [45](#)
 - [ApplyDamage](#), [48](#)
 - [AttackEnter](#), [49](#)
 - [AttackPlayer](#), [49](#)
 - [CanSee](#), [49](#)
 - [ChaseEnter](#), [49](#)
 - [ChasePlayer](#), [50](#)
 - [CollisionAvoidance](#), [50](#)
 - [HasCollided](#), [50](#)
 - [Investigating](#), [50](#)
 - [Movement](#), [50](#)
 - [Seek](#), [51](#)
 - [SetCurrentState](#), [51](#)
 - [SetPath](#), [51](#)
 - [SetPathNodes](#), [52](#)
 - [Update](#), [52](#)
- [CAIController.cpp](#), [260](#)

- CAIController.h, 261, 262
- CAINode.h, 199, 200
- CalculateCost
 - Pathfinding, 162
- CalculatePath
 - Pathfinding, 162
- CameraManager, 52
 - AddCamera, 53
 - GetAllCameras, 53
 - GetRenderingCamera, 53
 - RemoveCamera, 53
 - SetRenderingCamera, 54
- CameraManager.h, 233, 234
- CAnimationSpriteComponent, 54
 - SetAnimationRectPosition, 55
 - SetAnimationRectSize, 55
 - Update, 55
- CAnimationSpriteComponent.h, 204, 205
- CanSee
 - CAIController, 49
- CAudio, 56
- CAudio.h, 232
- CAudioEmitterComponent, 56
 - Draw, 57
 - Load, 57, 58
 - Play, 58
 - SetRange, 58
 - Update, 58
- CAudioEmitterComponent.h, 205, 206
- CCamera, 59
 - Update, 59
- CCamera.h, 221
- CCameraComponent, 60
 - Draw, 60
 - getAttachedToParent, 61
 - GetPosition, 61
 - GetProjectionMatrix, 61
 - GetViewMatrix, 61
 - GetZoomLevel, 62
 - SetAttachedToParent, 62
 - SetZoomLevel, 62
 - Update, 63
- CCameraComponent.h, 206, 207
- CCharacter, 63
 - ApplyDamage, 64
 - Update, 64
- CCharacter.cpp, 269
- CCharacter.h, 269
- CComponent, 65
 - Draw, 66
 - GetTransform, 66
 - SetAnchor, 66
 - SetUseTranslucency, 67
 - Update, 67
- CComponent.h, 201, 202
- CellData, 67
- CEmitter, 68
- CEmitter.h, 232, 233
- CEntity, 68
 - HasCollided, 69
 - SetIsUI, 69
 - Update, 70
- CEntity.h, 203
- CGridCursor, 70
 - Update, 71
- CGridCursor.h, 215
- ChaseEnter
 - CAIController, 49
- ChasePlayer
 - AlarmEnemy, 32
 - CAIController, 50
 - DogEnemy, 138
 - GruntEnemy, 144
- ChaseState, 71
 - Enter, 72
 - Exit, 72
 - Update, 72
- CInteractable, 73
 - GetLastCollidedObject, 73
 - GetSprite, 74
 - HasCollided, 74
 - OnInteract, 74
 - SetInteractRange, 74
 - SetTexture, 75
 - SetTextureWIC, 75
 - Update, 75
- CInteractable.h, 270, 271
- CMaterial, 76
- CMaterial.h, 222
- CMesh, 76
- CMesh.h, 223
- CollisionAvoidance
 - CAIController, 50
- CollisionComponent, 77
 - Resolve, 77
- CollisionComponent.h, 234
- ConstantBuffer, 78
- CParticle, 78
 - Draw, 79
 - GetDirection, 79
 - GetLifetime, 79
 - getSpriteComponent, 80
 - GetVelocity, 80
 - SetDirection, 80
 - SetLifetime, 80
 - SetVelocity, 81
 - Update, 81
- CParticle.cpp, 214
- CParticle.h, 214
- CParticleEmitter, 81
 - Draw, 82
 - GetDirection, 83
 - GetLifetime, 83
 - GetVelocity, 83
 - SetDirection, 84
 - SetLifetime, 84

- SetSize, [84](#)
- SetTexture, [85](#)
- SetVelocity, [85](#)
- Update, [85](#)
- UseRandomDirection, [85](#)
- UseRandomLifetime, [86](#)
- UseRandomVelocity, [86](#)
- CParticleEmitter.h, [208](#)
- CPlayer, [87](#)
 - Update, [87](#)
- CPlayer.h, [271](#)
- CPlayerController, [87](#)
 - HandleInput, [88](#)
 - OnPossess, [88](#)
 - OnUnpossess, [88](#)
- CPlayerController.cpp, [271](#)
- CPlayerController.h, [272](#)
- CRigidBodyComponent, [89](#)
 - Draw, [89](#)
 - GetAcceleration, [89](#)
 - GetVelocity, [90](#)
 - SetAcceleration, [90](#)
 - SetVelocity, [90](#)
 - Update, [91](#)
- CRigidBodyComponent.cpp, [209](#)
- CRigidBodyComponent.h, [210](#)
- Crossbow, [91](#)
 - Update, [91](#)
- Crossbow.cpp, [297](#)
- Crossbow.h, [298](#)
- CSpriteComponent, [93](#)
 - Draw, [94](#)
 - GetTransform, [94](#)
 - LoadTexture, [94](#)
 - LoadTextureWIC, [94](#)
 - SetRenderRect, [94](#)
 - SetSpriteSize, [95](#)
 - SetTextureOffset, [95](#)
 - SetUseTranslucency, [95](#)
 - Update, [95](#)
- CSpriteComponent.h, [210](#), [211](#)
- CT_EditorEntity, [96](#)
 - Update, [96](#)
- CT_EditorEntity.h, [253](#)
- CT_EditorEntity_Enemy, [97](#)
 - InitialiseEntity, [98](#)
 - Update, [98](#)
- CT_EditorEntity_PlayerStart, [98](#)
 - Update, [99](#)
- CT_EditorEntity_Waypoint, [99](#)
 - InitialiseEntity, [100](#)
 - Update, [100](#)
- CT_EditorEntity_WeaponHolder, [100](#)
 - InitialiseEntity, [101](#)
 - Update, [101](#)
- CT_EditorGrid, [102](#)
 - Update, [102](#)
- CT_EditorGrid.h, [255](#)
- CT_EditorMain, [103](#)
- CT_EditorMain.h, [256](#)
- CT_EditorWindows, [103](#)
- CT_EditorWindows.h, [256](#)
- CT_PropData, [103](#)
- CTextRenderComponent, [104](#)
 - Draw, [105](#)
 - SetCharacterSize, [105](#)
 - SetJustification, [105](#)
 - SetReserveCount, [105](#)
 - SetSpriteSheetColumnsCount, [105](#)
 - Update, [106](#)
- CTextRenderComponent.h, [211](#), [212](#)
 - TextJustification, [212](#)
- CTexture, [106](#)
- CTexture.h, [224](#)
- CTile, [107](#)
 - Update, [107](#)
- CTile.h, [215](#)
- CTransform, [108](#)
- CTransform.h, [235](#), [236](#)
- CUIManager, [109](#)
- CUIManager.h, [236](#)
- CursorEntity, [110](#)
 - Update, [110](#)
- CursorEntity.h, [258](#)
- CWidget, [111](#)
 - AddChild, [111](#)
 - SetVisibility, [112](#)
 - SetWidgetTransform, [112](#)
- CWidget.cpp, [225](#)
- CWidget.h, [225](#)
- CWidget_Button, [112](#)
 - Bind_HoverEnd, [113](#)
 - Bind_HoverStart, [114](#)
 - Bind_OnButtonPressed, [114](#)
 - Bind_OnButtonReleased, [114](#)
 - SetButtonSize, [115](#)
 - SetText, [115](#)
 - SetTexture, [115](#)
 - SetVisibility, [115](#)
 - SetWidgetTransform, [116](#)
 - Update, [116](#)
- CWidget_Button.cpp, [225](#)
- CWidget_Button.h, [226](#)
- CWidget_Canvas, [117](#)
 - GetMousePosition, [117](#)
 - InitialiseCanvas, [118](#)
 - SetVisibility, [118](#)
 - Update, [118](#)
- CWidget_Canvas.h, [227](#), [228](#)
- CWidget_Image, [119](#)
 - SetVisibility, [119](#)
 - SetWidgetTransform, [120](#)
 - Update, [120](#)
- CWidget_Image.h, [228](#), [229](#)
- CWidget_Text, [121](#)
 - SetVisibility, [121](#)

- SetWidgetTransform, 122
- Update, 122
- CWidget_Text.h, 229
- CWorld, 122
 - mapSize, 123
- CWorld.h, 216
- CWorld_Edit.h, 218
- CWorld_Editable, 124
 - LoadWorld, 125
 - SetupWorld, 125
 - UnloadWorld, 125
- CWorld_Game, 125
 - CWorld_Game, 126
 - LoadEntities, 126
 - ReloadWorld, 126
 - SetupWorld, 126
 - UnloadWorld, 127
- CWorld_Game.h, 259
- CWorld_Menu, 127
- CWorld_Menu.h, 259
- CWorldManager, 127
 - LoadWorld, 128
- CWorldManager.h, 236
- Dagger, 129
- Dagger.h, 293, 294
- Debug, 129
 - GetLogging, 130
 - getOutput, 130
 - GetVisibility, 130
 - Log, 130
 - LogError, 131
 - LogHResult, 131
 - SetLogging, 131
 - SetVisibility, 132
- Debug.h, 237, 238
- DebugOutput, 132
- DebugOutput.h, 240
- DestroyAudio
 - AudioController, 39
- Dialogue, 132
- Dialogue.h, 272
- DialogueHandler, 133
 - AdvanceDialogue, 133
 - LoadDialogue, 133
 - SetDialogue, 134
- DialogueHandler.cpp, 273
- DialogueHandler.h, 273
- DialogueUI, 134
 - Advance, 135
 - SetName, 135
 - SetText, 136
 - ToggleDrawing, 136
 - Update, 136
- DialogueUI.cpp, 273
- DialogueUI.h, 274
- DidItHit
 - Projectile, 177
- DogEnemy, 137
 - AttackEnter, 137
 - AttackPlayer, 138
 - ChasePlayer, 138
 - OnDeath, 138
 - OnHit, 138
 - Update, 139
- DogEnemy.cpp, 263
- DogEnemy.h, 264
- Draw
 - CAudioEmitterComponent, 57
 - CCameraComponent, 60
 - CComponent, 66
 - CParticle, 79
 - CParticleEmitter, 82
 - CRigidBodyComponent, 89
 - CSpriteComponent, 94
 - CTextRenderComponent, 105
 - Weapon, 191
 - WeaponInterface, 193
- Engine, 139
- Engine.h, 213
- Enter
 - AttackState, 37
 - ChaseState, 72
 - InvestigateState, 148
 - PatrolState, 166
 - SearchState, 181
- EntityManager, 140
 - RemoveComponent, 140
 - RemoveEntity, 140
 - SortTranslucentComponents, 141
- EntityManager.h, 241, 242
- EventSystem, 141
 - AddListener, 141
 - TriggerEvent, 142
- EventSystem.h, 242, 243
- Exit
 - AttackState, 37
 - ChaseState, 72
 - InvestigateState, 149
 - PatrolState, 166
 - SearchState, 181
- FindClosestPatrolNode
 - Pathfinding, 162
- FindClosestWaypoint
 - Pathfinding, 164
- FindExtension
 - IO, 150
- Fireball, 142
- Fireball.cpp, 299
- Fireball.h, 299, 300
- FloatToStringWithDigits
 - Math, 157
- FromScreenToWorld
 - Math, 157
- GetAcceleration

- CRigidBodyComponent, 89
- GetAllCameras
 - CameraManager, 53
- GetAllEmittersWithinRange
 - AudioController, 39
- getAttachedToParent
 - CCameraComponent, 61
- GetAudio
 - AssetManager, 34
- GetDefaultMesh
 - AssetManager, 35
- GetDirection
 - CParticle, 79
 - CParticleEmitter, 83
- GetLastCollidedObject
 - CInteractable, 73
- GetLifetime
 - CParticle, 79
 - CParticleEmitter, 83
- GetLogging
 - Debug, 130
- GetMesh
 - AssetManager, 35
- GetMousePosition
 - CWidget_Canvas, 117
- getOutput
 - Debug, 130
- GetPathNodes
 - Pathfinding, 164
- GetPosition
 - CCameraComponent, 61
- GetProjectionMatrix
 - CCameraComponent, 61
- GetRenderingCamera
 - CameraManager, 53
- GetSprite
 - CInteractable, 74
- getSpriteComponent
 - CParticle, 80
- GetTexture
 - AssetManager, 35
- GetTextureWIC
 - AssetManager, 36
- GetTransform
 - CComponent, 66
 - CSpriteComponent, 94
- GetVelocity
 - CParticle, 80
 - CParticleEmitter, 83
 - CRigidBodyComponent, 90
- GetViewMatrix
 - CCameraComponent, 61
- GetVisibility
 - Debug, 130
- GetZoomLevel
 - CCameraComponent, 62
- GruntEnemy, 143
 - AttackPlayer, 143
 - ChasePlayer, 144
 - OnDeath, 144
 - OnHit, 144
 - Update, 144
- GruntEnemy.cpp, 265
- GruntEnemy.h, 265, 266
- HandleInput
 - CPlayerController, 88
 - PlayerController, 175
- HasCollided
 - CAIController, 50
 - CEntity, 69
 - CInteractable, 74
- HomingProjectile, 145
 - Update, 145
- HomingProjectile.cpp, 283
- HomingProjectile.h, 283, 284
- IInputable, 146
 - PressedDrop, 146
 - PressedHorizontal, 146
 - PressedInteract, 146
 - PressedVertical, 146
- IInputable.h, 221
- InitialiseCanvas
 - CWidget_Canvas, 118
- InitialiseEntity
 - CT_EditorEntity_Enemy, 98
 - CT_EditorEntity_Waypoint, 100
 - CT_EditorEntity_WeaponHolder, 101
- InputManager, 147
- InputManager.cpp, 243
- InputManager.h, 244
- IntToString
 - Math, 158
- InvestigateState, 148
 - Enter, 148
 - Exit, 149
 - Update, 149
- Investigating
 - CAIController, 50
- InvisibilityScroll, 149
- InvisibilityScroll.h, 297
- IO, 150
 - FindExtension, 150
- IO.h, 246
- IUsePickup, 150
 - UsePickup, 151
- IUsePickup.h, 274
- LevelCompleteMenu, 151
- LevelCompleteMenu.cpp, 284
- LevelCompleteMenu.h, 285
- LevelSelectMenu, 152
- LevelSelectMenu.cpp, 285
- LevelSelectMenu.h, 286, 287
- LevelTransporter, 153
 - OnInteract, 153

- LevelTransporter.h, 275
- Load
 - AudioEmitterEntity, 43
 - CAudioEmitterComponent, 57, 58
- LoadAudio
 - AudioController, 40
- LoadDialogue
 - DialogueHandler, 133
- LoadEntities
 - CWorld_Game, 126
- LoadTexture
 - CSpriteComponent, 94
- LoadTextureWIC
 - CSpriteComponent, 94
- LoadWorld
 - CWorld_Editable, 125
 - CWorldManager, 128
- Log
 - Debug, 130
- LogError
 - Debug, 131
- LogHResult
 - Debug, 131
- Longsword, 154
 - OnFire, 154
- Longsword.h, 294, 295
- MagicMissile, 155
 - OnFire, 155
- MagicMissile.cpp, 300
- MagicMissile.h, 300, 301
- MainMenu, 156
- MainMenu.cpp, 287
- MainMenu.h, 288
- mapSize
 - CWorld, 123
- MaterialPropertiesConstantBuffer, 156
- Math, 156
 - FloatToStringWithDigits, 157
 - FromScreenToWorld, 157
 - IntToString, 158
- Math.h, 247
- MeleeWeapon, 158
 - OnFire, 159
- MeleeWeapon.cpp, 296
- MeleeWeapon.h, 296
- Movement
 - CAIController, 50
- NecrodoggieconPage, 160
 - OnInteract, 160
- NecrodoggieconPage.h, 275
- OnDeath
 - AlarmEnemy, 32
 - DogEnemy, 138
 - GruntEnemy, 144
- OnFire
 - Longsword, 154
- MagicMissile, 155
- MeleeWeapon, 159
- Pickup, 168
- RangeWeapon, 179
- Weapon, 191
- WeaponInterface, 193
- OnHit
 - AlarmEnemy, 32
 - DogEnemy, 138
 - GruntEnemy, 144
- OnInteract
 - CInteractable, 74
 - LevelTransporter, 153
 - NecrodoggieconPage, 160
 - WeaponPickup< T >, 195
- OnPossess
 - CPlayerController, 88
 - PlayerController, 176
- OnUnpossess
 - CPlayerController, 88
 - PlayerController, 176
- Pathfinding, 161
 - CalculateCost, 162
 - CalculatePath, 162
 - FindClosestPatrolNode, 162
 - FindClosestWaypoint, 164
 - GetPathNodes, 164
 - Pathfinding, 161
 - SetPath, 164
 - SetPatrolNodes, 165
- Pathfinding.cpp, 200
- Pathfinding.h, 200, 201
- PatrolNode, 165
- PatrolState, 166
 - Enter, 166
 - Exit, 166
 - Update, 166
- PauseMenu, 167
 - Update, 167
- PauseMenu.cpp, 289
- PauseMenu.h, 289, 290
- Pickup, 168
 - OnFire, 168
 - Update, 170
- Pickup.cpp, 296
- Pickup.h, 297
- Play
 - CAudioEmitterComponent, 58
- PlayAudio
 - AudioController, 40
 - AudioEmitterEntity, 43, 44
- PlayerCharacter, 170
 - ApplyDamage, 172
 - Attack, 172
 - PressedDrop, 172
 - PressedHorizontal, 173
 - PressedInteract, 173
 - PressedUse, 173

- PressedVertical, [173](#)
- ToggleVisibility, [174](#)
- Update, [174](#)
- UsePickup, [174](#)
- PlayerCharacter.h, [275](#)
- PlayerController, [175](#)
 - HandleInput, [175](#)
 - OnPossess, [176](#)
 - OnUnpossess, [176](#)
 - Update, [176](#)
- PlayerController.h, [276](#)
- PlayMusic
 - SoundManager, [185](#)
- PlaySound
 - SoundManager, [185](#)
- PressedDrop
 - IInputable, [146](#)
 - PlayerCharacter, [172](#)
- PressedHorizontal
 - IInputable, [146](#)
 - PlayerCharacter, [173](#)
- PressedInteract
 - IInputable, [146](#)
 - PlayerCharacter, [173](#)
- PressedUse
 - PlayerCharacter, [173](#)
- PressedVertical
 - IInputable, [146](#)
 - PlayerCharacter, [173](#)
- Projectile, [177](#)
 - DidItHit, [177](#)
 - StartUp, [178](#)
 - Update, [178](#)
- Projectile.cpp, [290](#)
- Projectile.h, [290](#), [291](#)
- PropData, [178](#)
- RangeWeapon, [179](#)
 - OnFire, [179](#)
- RangeWeapon.h, [301](#)
- Rapier, [180](#)
- Rapier.h, [295](#)
- ReloadWorld
 - CWorld_Game, [126](#)
- RemoveAudio
 - AssetManager, [36](#)
- RemoveCamera
 - CameraManager, [53](#)
- RemoveComponent
 - EntityManager, [140](#)
- RemoveEmitter
 - AudioController, [41](#)
- RemoveEntity
 - EntityManager, [140](#)
- Resolve
 - CollisionComponent, [27](#)
- Resource.h, [252](#)
- resource.h, [252](#)
- SearchState, [180](#)
 - Enter, [181](#)
 - Exit, [181](#)
 - Update, [181](#)
- Seek
 - CAIController, [51](#)
- SetAcceleration
 - CRigidBodyComponent, [90](#)
- SetAnchor
 - CComponent, [66](#)
- SetAnimationRectPosition
 - CAnimationSpriteComponent, [55](#)
- SetAnimationRectSize
 - CAnimationSpriteComponent, [55](#)
- SetAttachedToParent
 - CCameraComponent, [62](#)
- SetAudio
 - AudioEmitterEntity, [44](#)
- SetButtonSize
 - CWidget_Button, [115](#)
- SetCharacterSize
 - CTextRenderComponent, [105](#)
- SetCurrentState
 - CAIController, [51](#)
- SetDialogue
 - DialogueHandler, [134](#)
- SetDirection
 - CParticle, [80](#)
 - CParticleEmitter, [84](#)
- SetInteractRange
 - CInteractable, [74](#)
- SetIsUI
 - CEntity, [69](#)
- SetJustification
 - CTextRenderComponent, [105](#)
- SetLifetime
 - CParticle, [80](#)
 - CParticleEmitter, [84](#)
- SetLogging
 - Debug, [131](#)
- SetName
 - DialogueUI, [135](#)
- SetPath
 - CAIController, [51](#)
 - Pathfinding, [164](#)
- SetPathNodes
 - CAIController, [52](#)
- SetPatrolNodes
 - Pathfinding, [165](#)
- SetRange
 - AudioEmitterEntity, [45](#)
 - CAudioEmitterComponent, [58](#)
- SetRenderingCamera
 - CameraManager, [54](#)
- SetRenderRect
 - CSpriteComponent, [94](#)
- SetReserveCount
 - CTextRenderComponent, [105](#)

- SetSize
 - CParticleEmitter, 84
- SetSpriteSheetColumnsCount
 - CTextRenderComponent, 105
- SetSpriteSize
 - CSpriteComponent, 95
- SetText
 - CWidget_Button, 115
 - DialogueUI, 136
- SetTexture
 - CInteractable, 75
 - CParticleEmitter, 85
 - CWidget_Button, 115
- SetTextureOffset
 - CSpriteComponent, 95
- SetTextureWIC
 - CInteractable, 75
- SettingsMenu, 182
 - Update, 182
- SettingsMenu.cpp, 292
- SettingsMenu.h, 292, 293
- SetupWorld
 - CWorld_Editable, 125
 - CWorld_Game, 126
- SetUserType
 - WeaponInterface, 194
- SetUseTranslucency
 - CComponent, 67
 - CSpriteComponent, 95
- SetVelocity
 - CParticle, 81
 - CParticleEmitter, 85
 - CRigidBodyComponent, 90
- SetVisibility
 - CWidget, 112
 - CWidget_Button, 115
 - CWidget_Canvas, 118
 - CWidget_Image, 119
 - CWidget_Text, 121
 - Debug, 132
- SetWeapon
 - Weapon, 192
 - WeaponInterface, 194
 - WeaponPickup< T >, 195
- SetWidgetTransform
 - CWidget, 112
 - CWidget_Button, 116
 - CWidget_Image, 120
 - CWidget_Text, 122
- SetZoomLevel
 - CCameraComponent, 62
- ShieldScroll, 183
- ShieldScroll.h, 297
- SimpleVertex, 183
- SortTranslucentComponents
 - EntityManager, 141
- SoundManager, 183
 - AddSound, 184
 - PlayMusic, 185
 - PlaySound, 185
- SoundManager.cpp, 276
- SoundManager.h, 277
- StartUp
 - Projectile, 178
- State, 185
- State.cpp, 266
- State.h, 266, 267
- StopAudio
 - AudioController, 41
- structures.h, 224
- TestUI, 186
 - Update, 186
- TestUI.h, 277
- TextJustification
 - CTextRenderComponent.h, 212
- ToggleDrawing
 - DialogueUI, 136
- ToggleVisibility
 - PlayerCharacter, 174
- TriggerEvent
 - EventSystem, 142
- UnloadWorld
 - CWorld_Editable, 125
 - CWorld_Game, 127
- Update
 - AlarmEnemy, 33
 - AttackState, 37
 - AudioEmitterEntity, 45
 - CAIController, 52
 - CAnimationSpriteComponent, 55
 - CAudioEmitterComponent, 58
 - CCamera, 59
 - CCameraComponent, 63
 - CCharacter, 64
 - CComponent, 67
 - CEntity, 70
 - CGridCursor, 71
 - ChaseState, 72
 - CInteractable, 75
 - CParticle, 81
 - CParticleEmitter, 85
 - CPlayer, 87
 - CRigidBodyComponent, 91
 - Crossbow, 91
 - CSpriteComponent, 95
 - CT_EditorEntity, 96
 - CT_EditorEntity_Enemy, 98
 - CT_EditorEntity_PlayerStart, 99
 - CT_EditorEntity_Waypoint, 100
 - CT_EditorEntity_WeaponHolder, 101
 - CT_EditorGrid, 102
 - CTextRenderComponent, 106
 - CTile, 107
 - CursorEntity, 110
 - CWidget_Button, 116

- CWidget_Canvas, [118](#)
- CWidget_Image, [120](#)
- CWidget_Text, [122](#)
- DialogueUI, [136](#)
- DogEnemy, [139](#)
- GruntEnemy, [144](#)
- HomingProjectile, [145](#)
- InvestigateState, [149](#)
- PatrolState, [166](#)
- PauseMenu, [167](#)
- Pickup, [170](#)
- PlayerCharacter, [174](#)
- PlayerController, [176](#)
- Projectile, [178](#)
- SearchState, [181](#)
- SettingsMenu, [182](#)
- TestUI, [186](#)
- Weapon, [192](#)
- WeaponInterface, [194](#)
- weaponUI, [196](#)
- updateUI
 - weaponUI, [197](#)
- UsePickup
 - IUsePickup, [151](#)
 - PlayerCharacter, [174](#)
- UseRandomDirection
 - CParticleEmitter, [85](#)
- UseRandomLifetime
 - CParticleEmitter, [86](#)
- UseRandomVelocity
 - CParticleEmitter, [86](#)
- Vector2Base< T >, [187](#)
- Vector3.h, [248](#)
- Vector3Base< T >, [188](#)
- WaypointNode, [189](#)
- Weapon, [190](#)
 - Draw, [191](#)
 - OnFire, [191](#)
 - SetWeapon, [192](#)
 - Update, [192](#)
- WeaponInterface, [192](#)
 - Draw, [193](#)
 - OnFire, [193](#)
 - SetUserType, [194](#)
 - SetWeapon, [194](#)
 - Update, [194](#)
- WeaponInterface.h, [278](#)
- WeaponPickup< T >, [195](#)
 - OnInteract, [195](#)
 - SetWeapon, [195](#)
- WeaponPickup.h, [279](#)
- weapons.h, [281](#), [282](#)
- weaponUI, [196](#)
 - Update, [196](#)
 - updateUI, [197](#)
- weaponUI.cpp, [301](#)
- weaponUI.h, [302](#)
- WorldConstants.h, [257](#)