Anfaengerpraktikum

v.0.1.0

Generated by Doxygen 1.12.0

1 anfaengerpraktikum 1

2 Third-Party Notices 3

2.1 License Texts . . . 3

2.1.1 godotengine/godot-cpp . . . 3

2.1.2 libgit2/libgit2 . . . 4

2.1.3 libssh2/libssh2 . . . 17

2.1.4 OpenSSL . . . 17

3 Namespace Index 21

3.1 Namespace List . . . 21

4 Hierarchical Index 23

4.1 Class Hierarchy . . . 23

5 Class Index 25

5.1 Class List . . . 25

6 File Index 27

6.1 File List . . . 27

7 Namespace Documentation 29

7.1 GdMUT Namespace Reference . . . 29

7.2 GdMUT.Components Namespace Reference . . . 29

8 Class Documentation 31

8.1 BaseEnemy Class Reference . . . 31

8.1.1 Detailed Description . . . 33

8.1.2 Member Enumeration Documentation . . . 33

8.1.2.1 State . . . 33

8.1.3 Member Function Documentation . . . 33

8.1.3.1 _Process() . . . 33

8.1.3.2 _Ready() . . . 34

8.1.3.3 CheckLineOfSight() . . . 34

8.1.3.4 CheckPlayerHit() . . . 35

8.1.3.5 Die() . . . 36

8.1.3.6 FlipRotation() . . . 36

8.1.3.7 HandleMovement() . . . 36

8.1.3.8 IsCloseTo() . . . 37

8.1.3.9 IsDead() . . . 39

8.1.3.10 OnDetectionBodyEntered() . . . 39

8.1.3.11 OnHitboxAreaEntered() . . . 39

8.1.3.12 OnPursuingRadiusBodyExited() . . . 40

8.1.3.13 OnSwordHitBoxBodyEntered() . . . 40

8.1.3.14 Respawn() . . . 41

8.1.3.15 SetRotation() . . . 41
8.1.3.16 TakeDamage() . . . 41
8.1.3.17 UpdateAnimation() . . . 42
8.1.4 Member Data Documentation . . . 42
8.1.4.1 AlreadyHit . . . 42
8.1.4.2 AnimationState . . . 43
8.1.4.3 Armor . . . 43
8.1.4.4 CollisionPolygon . . . 43
8.1.4.5 CurrentHealthPoints . . . 43
8.1.4.6 CurrentStamina . . . 43
8.1.4.7 CurrentTarget . . . 43
8.1.4.8 Damage . . . 44
8.1.4.9 Dead . . . 44
8.1.4.10 FrontCollisionRayCast . . . 44
8.1.4.11 HealthBar . . . 44
8.1.4.12 LeftFallProtection . . . 44
8.1.4.13 LineOfSight . . . 44
8.1.4.14 MainCollision . . . 45
8.1.4.15 MaxHealthPoints . . . 45
8.1.4.16 MaxStamina . . . 45
8.1.4.17 Player . . . 45
8.1.4.18 Pursuing . . . 45
8.1.4.19 Respawnable . . . 45
8.1.4.20 ReturnToStart . . . 46
8.1.4.21 ReturnToStartAfter . . . 46
8.1.4.22 RightFallProtection . . . 46
8.1.4.23 SinAmount . . . 46
8.1.4.24 Speed . . . 46
8.1.4.25 Sprite . . . 46
8.1.4.26 StartPosition . . . 47
8.1.4.27 StartRotation . . . 47
8.1.4.28 SwordHitbox . . . 47
8.1.4.29 TargetPosition . . . 47
8.1.5 Property Documentation . . . 47
8.1.5.1 Id . . . 47
8.2 BloodVial Class Reference . . . 48
8.2.1 Detailed Description . . . 48
8.2.2 Member Function Documentation . . . 48
8.2.2.1 _Ready() . . . 48
8.2.2.2 AddMaxUses() . . . 48
8.2.2.3 LevelHealAmount() . . . 49
8.2.2.4 ResetUses() . . . 49

8.2.2.5 UseBloodVial() . . . 49
8.3 Boss1 Class Reference . . . 50
8.3.1 Detailed Description . . . 52
8.3.2 Member Function Documentation . . . 52
8.3.2.1 _Process() . . . 52
8.3.2.2 _Ready() . . . 52
8.3.2.3 HandleRegeneration() . . . 52
8.3.2.4 ReviveEnemies() . . . 53
8.3.2.5 ShowPopupMessage() . . . 53
8.3.2.6 StartGlowing() . . . 54
8.3.3 Member Data Documentation . . . 54
8.3.3.1 EnemiesRevived . . . 54
8.3.3.2 RegenAmount . . . 55
8.3.3.3 RegenCooldown . . . 55
8.3.3.4 RegenTimer . . . 55
8.4 Checkpoint Class Reference . . . 55
8.4.1 Detailed Description . . . 56
8.4.2 Member Function Documentation . . . 56
8.4.2.1 _Ready() . . . 56
8.4.2.2 OnPlayerBodyEntered() . . . 56
8.4.3 Member Data Documentation . . . 56
8.4.3.1 Player . . . 56
8.5 Damage Class Reference . . . 57
8.5.1 Detailed Description . . . 57
8.5.2 Constructor & Destructor Documentation . . . 57
8.5.2.1 Damage() . . . 57
8.5.3 Member Function Documentation . . . 58
8.5.3.1 GetPhysicalDMG() . . . 58
8.5.3.2 GetPushAmount() . . . 58
8.5.3.3 GetSource() . . . 58
8.5.3.4 GetTrueDMG() . . . 59
8.5.4 Member Data Documentation . . . 59
8.5.4.1 PhysicalDMG . . . 59
8.5.4.2 PushAmount . . . 59
8.5.4.3 Source . . . 59
8.5.4.4 TrueDMG . . . 60
8.6 Door Class Reference . . . 60
8.6.1 Detailed Description . . . 61
8.6.2 Member Function Documentation . . . 61
8.6.2.1 _Ready() . . . 61
8.6.2.2 OnPlayerBodyEntered() . . . 61
8.6.3 Member Data Documentation . . . 61

8.6.3.1 Spawn . . . 61
8.6.4 Property Documentation . . . 62
8.6.4.1 DestinationDoorTag . . . 62
8.6.4.2 DestinationLevelTag . . . 62
8.6.4.3 SpawnDirection . . . 62
8.7 HealthBar Class Reference . . . 62
8.7.1 Detailed Description . . . 63
8.7.2 Member Function Documentation . . . 63
8.7.2.1 _Process() . . . 63
8.7.2.2 _Ready() . . . 63
8.8 Hud Class Reference . . . 64
8.8.1 Detailed Description . . . 64
8.8.2 Member Function Documentation . . . 64
8.8.2.1 _Process() . . . 64
8.8.2.2 _Ready() . . . 65
8.8.2.3 OnResumeButtonPressed() . . . 65
8.8.2.4 OnSaveButtonPressed() . . . 65
8.8.2.5 OnSaveMenuButtonPressed() . . . 66
8.8.2.6 OnSaveQuitButtonPressed() . . . 66
8.8.2.7 TogglePause() . . . 66
8.8.3 Member Data Documentation . . . 66
8.8.3.1 AnimationPlayer . . . 66
8.8.3.2 Buttons . . . 67
8.8.3.3 Enabled . . . 67
8.9 Interactable Class Reference . . . 67
8.9.1 Detailed Description . . . 68
8.9.2 Member Function Documentation . . . 68
8.9.2.1 _Process() . . . 68
8.9.2.2 _Ready() . . . 68
8.9.2.3 OnAreaBodyExited() . . . 68
8.9.3 Member Data Documentation . . . 69
8.9.3.1 Area . . . 69
8.9.3.2 Player . . . 69
8.9.3.3 PopUp . . . 69
8.9.3.4 TextLabel . . . 69
8.9.4 Property Documentation . . . 70
8.9.4.1 Text . . . 70
8.10 LevelManager Class Reference . . . 70
8.10.1 Detailed Description . . . 70
8.10.2 Member Function Documentation . . . 71
8.10.2.1 _Ready() . . . 71
8.10.2.2 OnLevelSpawn() . . . 71

8.11 MainMenu Class Reference . . . 72
8.11.1 Detailed Description . . . 73
8.11.2 Member Function Documentation . . . 73
8.11.2.1 _Ready() . . . 73
8.11.2.2 Change() . . . 74
8.11.2.3 OnBackButtonPressed() . . . 74
8.11.2.4 OnContinueButtonPressed() . . . 75
8.11.2.5 OnDeleteConfirmationCanceled() . . . 75
8.11.2.6 OnDeleteConfirmationCloseRequested() . . . 75
8.11.2.7 OnDeleteConfirmationConfirmed() . . . 75
8.11.2.8 OnLoadGameButtonPressed() . . . 76
8.11.2.9 OnNewGameButtonPressed() . . . 76
8.11.2.10 OnQuitButtonPressed() . . . 76
8.11.2.11 OnSave1DeletePressed() . . . 76
8.11.2.12 OnSave1SelectPressed() . . . 77
8.11.2.13 OnSave2DeletePressed() . . . 77
8.11.2.14 OnSave2SelectPressed() . . . 77
8.11.2.15 OnSave3DeletePressed() . . . 78
8.11.2.16 OnSave3SelectPressed() . . . 78
8.11.3 Member Data Documentation . . . 78
8.11.3.1 ContinueButton . . . 78
8.11.3.2 DeleteButton . . . 78
8.11.3.3 DeleteConfirmation . . . 79
8.11.3.4 InfoLabel . . . 79
8.11.3.5 MenuState . . . 79
8.11.3.6 Navigation . . . 79
8.11.3.7 SaveLabel . . . 79
8.11.3.8 SavesContainer . . . 79
8.11.3.9 SaveToDelete . . . 80
8.11.3.10 SelectButton . . . 80
8.12 MainMenuBackground Class Reference . . . 80
8.12.1 Detailed Description . . . 80
8.12.2 Member Function Documentation . . . 80
8.12.2.1 _Process() . . . 80
8.12.3 Member Data Documentation . . . 81
8.12.3.1 ScrollSpeed . . . 81
8.13 NavigationManager Class Reference . . . 81
8.13.1 Detailed Description . . . 82
8.13.2 Member Function Documentation . . . 82
8.13.2.1 _Ready() . . . 82
8.13.2.2 DeferredChangeScene() . . . 82
8.13.2.3 GoToLevel() . . . 83

8.13.2.4 OnTriggerPlayerSpawnEventHandler() . . . 84
8.13.2.5 TriggerPlayerSpawn() . . . 84
8.13.3 Member Data Documentation . . . 84
8.13.3.1 SceneBoss . . . 84
8.13.3.2 SceneIntro . . . 84
8.13.3.3 SceneLevel1 . . . 85
8.13.3.4 SceneLevelOne . . . 85
8.13.3.5 SceneLevelTwo . . . 85
8.13.3.6 SceneMainMenu . . . 85
8.13.4 Property Documentation . . . 85
8.13.4.1 Instance . . . 85
8.13.4.2 SpawnDoorTag . . . 86
8.14 Player Class Reference . . . 86
8.14.1 Detailed Description . . . 88
8.14.2 Member Function Documentation . . . 88
8.14.2.1 _PhysicsProcess() . . . 88
8.14.2.2 _Ready() . . . 88
8.14.2.3 CreateDashEffect() . . . 89
8.14.2.4 DashInProgress() . . . 89
8.14.2.5 GetBloodVials() . . . 90
8.14.2.6 GetDamage() . . . 90
8.14.2.7 HandleJump() . . . 91
8.14.2.8 HandleMovement() . . . 91
8.14.2.9 IsAttacking() . . . 92
8.14.2.10 IsBlocking() . . . 92
8.14.2.11 MaxHeal() . . . 93
8.14.2.12 OnSpawn() . . . 93
8.14.2.13 RegenerateStamina() . . . 93
8.14.2.14 Respawn() . . . 94
8.14.2.15 SetSinAmount() . . . 94
8.14.2.16 SlowPlayer() . . . 94
8.14.2.17 StartDash() . . . 95
8.14.2.18 StopDash() . . . 95
8.14.2.19 TakeDamage() . . . 95
8.14.2.20 UpdateAnimations() . . . 96
8.14.2.21 UseStamina() . . . 97
8.14.3 Member Data Documentation . . . 97
8.14.3.1 AnimationPlayer . . . 97
8.14.3.2 BloodVials . . . 97
8.14.3.3 CanDash . . . 98
8.14.3.4 DashDirection . . . 98
8.14.3.5 DashEffect . . . 98

8.14.3.6 DashSpeed . . . 98
8.14.3.7 DashTimer . . . 98
8.14.3.8 DashTrailInterval . . . 98
8.14.3.9 DashTrailTimer . . . 99
8.14.3.10 HauptHitbox . . . 99
8.14.3.11 IsDashing . . . 99
8.14.3.12 JUMP_VELOCITY . . . 99
8.14.3.13 JumpCount . . . 99
8.14.3.14 JumpMax . . . 99
8.14.3.15 LastAttack . . . 100
8.14.3.16 PlayerHitbox . . . 100
8.14.3.17 SinDisplay . . . 100
8.14.3.18 SPEED . . . 100
8.14.3.19 Sprite . . . 100
8.14.3.20 SwordCollision . . . 100
8.14.3.21 TimeSinceLastStaminaUse . . . 101
8.15 PlayerStats Class Reference . . . 101
8.15.1 Detailed Description . . . 103
8.15.2 Member Function Documentation . . . 103
8.15.2.1 _Ready() . . . 103
8.15.2.2 GetBVCurrentUses() . . . 103
8.15.2.3 GetBVHealAmount() . . . 103
8.15.2.4 GetBVMaxUses() . . . 104
8.15.2.5 GetCurrentHealth() . . . 104
8.15.2.6 GetCurrentLevelTag() . . . 104
8.15.2.7 GetMaxHealthPoints() . . . 105
8.15.2.8 GetMaxStamina() . . . 105
8.15.2.9 GetPosition() . . . 105
8.15.2.10 GetRespawnLevelTag() . . . 106
8.15.2.11 GetSinAmount() . . . 106
8.15.2.12 GetSpawnPoint() . . . 106
8.15.2.13 GetStamina() . . . 107
8.15.2.14 Reload() . . . 107
8.15.2.15 SetBVCurrentUses() . . . 107
8.15.2.16 SetBVHealAmount() . . . 107
8.15.2.17 SetBVMaxUses() . . . 108
8.15.2.18 SetCurrentHealth() . . . 108
8.15.2.19 SetCurrentLevelTag() . . . 108
8.15.2.20 SetMaxHealthPoints() . . . 109
8.15.2.21 SetMaxStamina() . . . 109
8.15.2.22 SetPosition() . . . 109
8.15.2.23 SetRespawnLevelTag() . . . 110

8.15.2.24 SetSinAmount() . . . 110
8.15.2.25 SetSpawnPoint() . . . 110
8.15.2.26 SetStamina() . . . 111
8.15.3 Member Data Documentation . . . 111
8.15.3.1 BVCurrentUses . . . 111
8.15.3.2 BVHealAmount . . . 111
8.15.3.3 BVMaxUses . . . 112
8.15.3.4 CurrentHealth . . . 112
8.15.3.5 CurrentLevelTag . . . 112
8.15.3.6 CurrentStamina . . . 112
8.15.3.7 MaxHealthPoints . . . 112
8.15.3.8 MaxStamina . . . 112
8.15.3.9 Position . . . 113
8.15.3.10 RespawnLevelTag . . . 113
8.15.3.11 SinAmount . . . 113
8.15.3.12 SpawnPoint . . . 113
8.15.4 Property Documentation . . . 113
8.15.4.1 Instance . . . 113
8.16 Spike Class Reference . . . 114
8.16.1 Detailed Description . . . 114
8.16.2 Member Function Documentation . . . 114
8.16.2.1 _Ready() . . . 114
8.16.2.2 GetDamage() . . . 115
8.16.2.3 OnPlayerBodyEntered() . . . 115
8.16.2.4 OnPlayerBodyExited() . . . 115
8.16.2.5 OnTimerTimeout() . . . 116
8.16.3 Member Data Documentation . . . 116
8.16.3.1 Damage . . . 116
8.16.3.2 Player . . . 116
8.17 SpikeDynamic Class Reference . . . 116
8.17.1 Detailed Description . . . 117
8.17.2 Member Function Documentation . . . 117
8.17.2.1 _Ready() . . . 117
8.17.2.2 GetDamage() . . . 118
8.17.2.3 OnPlayerBodyEntered() . . . 118
8.17.2.4 OnPlayerBodyExited() . . . 118
8.17.2.5 OnTimerTimeout() . . . 119
8.17.3 Member Data Documentation . . . 119
8.17.3.1 Damage . . . 119
8.17.3.2 Player . . . 119
8.18 StaminaBar Class Reference . . . 119
8.18.1 Detailed Description . . . 120

8.18.2 Member Function Documentation . . . 120
8.18.2.1 _Process() . . . 120
8.18.2.2 _Ready() . . . 120
8.19 StorageManager Class Reference . . . 121
8.19.1 Detailed Description . . . 122
8.19.2 Member Function Documentation . . . 122
8.19.2.1 _Ready() . . . 122
8.19.2.2 GetLastSaveId() . . . 122
8.19.2.3 GetSaves() . . . 122
8.19.2.4 LoadGameFile() . . . 122
8.19.2.5 LoadSettings() . . . 123
8.19.2.6 SaveAll() . . . 123
8.19.2.7 SaveGameFile() . . . 124
8.19.2.8 SaveSettings() . . . 124
8.19.2.9 SetLastSaveId() . . . 125
8.19.2.10 SetSaves() . . . 125
8.19.3 Member Data Documentation . . . 125
8.19.3.1 LastSaveId . . . 125
8.19.3.2 PathSave . . . 126
8.19.3.3 PathSettings . . . 126
8.19.3.4 Saves . . . 126
8.19.4 Property Documentation . . . 126
8.19.4.1 Instance . . . 126

9 File Documentation **127**

9.1 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/addons/↩ GDMUT/Dock.cs File Reference . . . 127
9.2 Dock.cs . . . 127
9.3 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/addons/↩ GDMUT/GDMUT.cs File Reference . . . 129
9.4 GDMUT.cs . . . 129
9.5 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/addons/↩ GDMUT/MethodResult.cs File Reference . . . 130
9.6 MethodResult.cs . . . 130
9.7 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/addons/↩ GDMUT/Result.cs File Reference . . . 131
9.8 Result.cs . . . 131
9.9 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/addons/↩ GDMUT/TestFunction.cs File Reference . . . 131
9.10 TestFunction.cs . . . 131
9.11 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/addons/↩ GDMUT/TestLoader.cs File Reference . . . 131
9.12 TestLoader.cs . . . 131

9.13 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/addons/↩ GDMUT/TestResult.cs File Reference . . . 133
9.14 TestResult.cs . . . 133
9.15 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/addons/godot-git-plugin/THIRDPARTY.md File Reference . . . 134
9.16 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/↩ README.md File Reference . . . 134
9.17 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/↩ BaseEnemy.cs File Reference . . . 134
9.18 BaseEnemy.cs . . . 134
9.19 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/↩ BloodVial.cs File Reference . . . 138
9.20 BloodVial.cs . . . 138
9.21 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/↩ Boss1.cs File Reference . . . 139
9.22 Boss1.cs . . . 139
9.23 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/↩ Checkpoint.cs File Reference . . . 140
9.24 Checkpoint.cs . . . 140
9.25 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/↩ Damage.cs File Reference . . . 141
9.26 Damage.cs . . . 141
9.27 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/↩ Door.cs File Reference . . . 141
9.28 Door.cs . . . 142
9.29 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/↩ HealthBar.cs File Reference . . . 142
9.30 HealthBar.cs . . . 142
9.31 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/↩ Hud.cs File Reference . . . 143
9.32 Hud.cs . . . 143
9.33 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/↩ Interactable.cs File Reference . . . 143
9.34 Interactable.cs . . . 144
9.35 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/↩ LevelManager.cs File Reference . . . 144
9.36 LevelManager.cs . . . 144
9.37 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/↩ MainMenu.cs File Reference . . . 145
9.38 MainMenu.cs . . . 145
9.39 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/↩ MainMenuBackground.cs File Reference . . . 147
9.40 MainMenuBackground.cs . . . 147
9.41 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/↩ NavigationManager.cs File Reference . . . 148
9.42 NavigationManager.cs . . . 148

9.43 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/↩ Player.cs File Reference 149
9.44 Player.cs 149
9.45 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/↩ PlayerStats.cs File Reference 153
9.46 PlayerStats.cs 154
9.47 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/↩ Spike.cs File Reference 155
9.48 Spike.cs 155
9.49 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/↩ SpikeDynamic.cs File Reference 156
9.50 SpikeDynamic.cs 156
9.51 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/↩ StaminaBar.cs File Reference 157
9.52 StaminaBar.cs 157
9.53 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/↩ StorageManager.cs File Reference 158
9.54 StorageManager.cs 158
9.55 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/Test↩ Class.cs File Reference 159
9.56 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/Test↩ Class.cs 159

Index **161**

Chapter 1

anfaengerpraktikum

Praktikum agile Computerspielentwicklung.

Chapter 2

Third-Party Notices

The Godot Git Plugin source code uses the following third-party source code:

1. godotengine/godot-cpp - MIT License - https://github.com/godotengine/godot-cpp/tree/02336831735
2. libgit2/libgit2 - GPLv2 with a special Linking Exception - https://github.com/libgit2/libgit2/tree/b7bad5
3. libssh2/libssh2 - BSD-3-Clause License - https://github.com/libssh2/libssh2/tree/635caa90787220ac

We also link to these third-party libraries (only in the compiled binary form):

1. OpenSSL - Only on Linux and MacOS - OpenSSL License - http://www.openssl.↩ org/source/openssl-1.1.1s.tar.gz

2.1 License Texts

2.1.1 godotengine/godot-cpp

```
# MIT License

Copyright (c) 2017-2022 Godot Engine contributors.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

2.1.2 libgit2/libgit2

```
 libgit2 is Copyright (C) the libgit2 contributors,
 unless otherwise stated. See the AUTHORS file for details.

 Note that the only valid version of the GPL as far as this project
 is concerned is _this_ particular version of the license (ie v2, not
 v2.2 or v3.x or whatever), unless explicitly otherwise stated.

----------------------------------------------------------------------

            LINKING EXCEPTION

 In addition to the permissions in the GNU General Public License,
 the authors give you unlimited permission to link the compiled
 version of this library into combinations with other programs,
 and to distribute those combinations without any restriction
 coming from the use of this file.  (The General Public License
 restrictions do apply in other respects; for example, they cover
 modification of the file, and distribution when not linked into
 a combined executable.)

----------------------------------------------------------------------

            GNU GENERAL PUBLIC LICENSE
               Version 2, June 1991

 Copyright (C) 1989, 1991 Free Software Foundation, Inc.
                       59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users.  This
General Public License applies to most of the Free Software
Foundation's software and to any other program whose authors commit to
using it.  (Some other Free Software Foundation software is covered by
the GNU Library General Public License instead.)  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
this service if you wish), that you receive source code or can get it
if you want it, that you can change the software or use pieces of it
in new free programs; and that you know you can do these things.

  To protect your rights, we need to make restrictions that forbid
anyone to deny you these rights or to ask you to surrender the rights.
These restrictions translate to certain responsibilities for you if you
distribute copies of the software, or if you modify it.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must give the recipients all the rights that
you have.  You must make sure that they, too, receive or can get the
source code.  And you must show them these terms so they know their
rights.

  We protect your rights with two steps: (1) copyright the software, and
(2) offer you this license which gives you legal permission to copy,
distribute and/or modify the software.

  Also, for each author's protection and ours, we want to make certain
that everyone understands that there is no warranty for this free
software.  If the software is modified by someone else and passed on, we
want its recipients to know that what they have is not the original, so
that any problems introduced by others will not reflect on the original
authors' reputations.

  Finally, any free program is threatened constantly by software
patents.  We wish to avoid the danger that redistributors of a free
program will individually obtain patent licenses, in effect making the
program proprietary.  To prevent this, we have made it clear that any
patent must be licensed for everyone's free use or not licensed at all.

  The precise terms and conditions for copying, distribution and
modification follow.

            GNU GENERAL PUBLIC LICENSE
   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

  0. This License applies to any program or other work which contains
a notice placed by the copyright holder saying it may be distributed
```

under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is

```
    allowed only for noncommercial distribution and only if you
    received the program in object code or executable form with such
    an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for
making modifications to it.  For an executable work, complete source
code means all the source code for all modules it contains, plus any
associated interface definition files, plus the scripts used to
control compilation and installation of the executable.  However, as a
special exception, the source code distributed need not include
anything that is normally distributed (in either source or binary
form) with the major components (compiler, kernel, and so on) of the
operating system on which the executable runs, unless that component
itself accompanies the executable.

If distribution of executable or object code is made by offering
access to copy from a designated place, then offering equivalent
access to copy the source code from the same place counts as
distribution of the source code, even though third parties are not
compelled to copy the source along with the object code.

  4. You may not copy, modify, sublicense, or distribute the Program
except as expressly provided under this License.  Any attempt
otherwise to copy, modify, sublicense or distribute the Program is
void, and will automatically terminate your rights under this License.
However, parties who have received copies, or rights, from you under
this License will not have their licenses terminated so long as such
parties remain in full compliance.

  5. You are not required to accept this License, since you have not
signed it.  However, nothing else grants you permission to modify or
distribute the Program or its derivative works.  These actions are
prohibited by law if you do not accept this License.  Therefore, by
modifying or distributing the Program (or any work based on the
Program), you indicate your acceptance of this License to do so, and
all its terms and conditions for copying, distributing or modifying
the Program or works based on it.

  6. Each time you redistribute the Program (or any work based on the
Program), the recipient automatically receives a license from the
original licensor to copy, distribute or modify the Program subject to
these terms and conditions.  You may not impose any further
restrictions on the recipients' exercise of the rights granted herein.
You are not responsible for enforcing compliance by third parties to
this License.

  7. If, as a consequence of a court judgment or allegation of patent
infringement or for any other reason (not limited to patent issues),
conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot
distribute so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you
may not distribute the Program at all.  For example, if a patent
license would not permit royalty-free redistribution of the Program by
all those who receive copies directly or indirectly through you, then
the only way you could satisfy both it and this License would be to
refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under
any particular circumstance, the balance of the section is intended to
apply and the section as a whole is intended to apply in other
circumstances.

It is not the purpose of this section to induce you to infringe any
patents or other property right claims or to contest validity of any
such claims; this section has the sole purpose of protecting the
integrity of the free software distribution system, which is
implemented by public license practices.  Many people have made
generous contributions to the wide range of software distributed
through that system in reliance on consistent application of that
system; it is up to the author/donor to decide if he or she is willing
to distribute software through any other system and a licensee cannot
impose that choice.

This section is intended to make thoroughly clear what is believed to
be a consequence of the rest of this License.

  8. If the distribution and/or use of the Program is restricted in
certain countries either by patents or by copyrighted interfaces, the
original copyright holder who places the Program under this License
may add an explicit geographical distribution limitation excluding
those countries, so that distribution is permitted only in or among
countries not thus excluded.  In such case, this License incorporates
the limitation as if written in the body of this License.

  9. The Free Software Foundation may publish revised and/or new versions
```

of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year>  <name of author>

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be

```
mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary.  Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the program
  `Gnomovision' (which makes passes at compilers) written by James Hacker.

  <signature of Ty Coon>, 1 April 1989
  Ty Coon, President of Vice

This General Public License does not permit incorporating your program into
proprietary programs.  If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library.  If this is what you want to do, use the GNU Library General
Public License instead of this License.

-----------------------------------------------------------------------

The bundled ZLib code is licensed under the ZLib license:

Copyright (C) 1995-2010 Jean-loup Gailly and Mark Adler

  This software is provided 'as-is', without any express or implied
  warranty.  In no event will the authors be held liable for any damages
  arising from the use of this software.

  Permission is granted to anyone to use this software for any purpose,
  including commercial applications, and to alter it and redistribute it
  freely, subject to the following restrictions:

  1. The origin of this software must not be misrepresented; you must not
     claim that you wrote the original software. If you use this software
     in a product, an acknowledgment in the product documentation would be
     appreciated but is not required.
  2. Altered source versions must be plainly marked as such, and must not be
     misrepresented as being the original software.
  3. This notice may not be removed or altered from any source distribution.

  Jean-loup Gailly        Mark Adler
  jloup@gzip.org          madler@alumni.caltech.edu

-----------------------------------------------------------------------

The Clar framework is licensed under the ISC license:

Copyright (c) 2011-2015 Vicent Marti

Permission to use, copy, modify, and/or distribute this software for any
purpose with or without fee is hereby granted, provided that the above
copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

-----------------------------------------------------------------------

The regex library (deps/regex/) is licensed under the GNU LGPL
(available at the end of this file).

Definitions for data structures and routines for the regular
expression library.

Copyright (C) 1985,1989-93,1995-98,2000,2001,2002,2003,2005,2006,2008
Free Software Foundation, Inc.
This file is part of the GNU C Library.

The GNU C Library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 2.1 of the License, or (at your option) any later version.

The GNU C Library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public
License along with the GNU C Library; if not, write to the Free
Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
```

02110-1301 USA.

The bundled winhttp definition files (deps/winhttp/) are licensed under
the GNU LGPL (available at the end of this file).

Copyright (C) 2007 Francois Gouget

This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public
License along with this library; if not, write to the Free Software
Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301, USA

GNU LESSER GENERAL PUBLIC LICENSE
Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts
as the successor of the GNU Library Public License, version 2, hence
the version number 2.1.]

Preamble

The licenses for most software are designed to take away your
freedom to share and change it. By contrast, the GNU General Public
Licenses are intended to guarantee your freedom to share and change
free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some
specially designated software packages--typically libraries--of the
Free Software Foundation and other authors who decide to use it. You
can use it too, but we suggest you first think carefully about whether
this license or the ordinary General Public License is the better
strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use,
not price. Our General Public Licenses are designed to make sure that
you have the freedom to distribute copies of free software (and charge
for this service if you wish); that you receive source code or can get
it if you want it; that you can change the software and use pieces of
it in new free programs; and that you are informed that you can do
these things.

To protect your rights, we need to make restrictions that forbid
distributors to deny you these rights or to ask you to surrender these
rights. These restrictions translate to certain responsibilities for
you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis
or for a fee, you must give the recipients all the rights that we gave
you. You must make sure that they, too, receive or can get the source
code. If you link other code with the library, you must provide
complete object files to the recipients, so that they can relink them
with the library after making changes to the library and recompiling
it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the
library, and (2) we offer you this license, which gives you legal
permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that
there is no warranty for the free library. Also, if the library is
modified by someone else and passed on, the recipients should know
that what they have is not the original version, so that the original
author's reputation will not be affected by problems that might be
introduced by others.

Finally, software patents pose a constant threat to the existence of
any free program. We wish to make sure that a company cannot
effectively restrict the users of a free program by obtaining a
restrictive license from a patent holder. Therefore, we insist that

```
any patent license obtained for a version of the library must be
consistent with the full freedom of use specified in this license.

  Most GNU software, including some libraries, is covered by the
ordinary GNU General Public License.  This license, the GNU Lesser
General Public License, applies to certain designated libraries, and
is quite different from the ordinary General Public License.  We use
this license for certain libraries in order to permit linking those
libraries into non-free programs.

  When a program is linked with a library, whether statically or using
a shared library, the combination of the two is legally speaking a
combined work, a derivative of the original library.  The ordinary
General Public License therefore permits such linking only if the
entire combination fits its criteria of freedom.  The Lesser General
Public License permits more lax criteria for linking other code with
the library.

  We call this license the "Lesser" General Public License because it
does Less to protect the user's freedom than the ordinary General
Public License.  It also provides other free software developers Less
of an advantage over competing non-free programs.  These disadvantages
are the reason we use the ordinary General Public License for many
libraries.  However, the Lesser license provides advantages in certain
special circumstances.

  For example, on rare occasions, there may be a special need to
encourage the widest possible use of a certain library, so that it becomes
a de-facto standard.  To achieve this, non-free programs must be
allowed to use the library.  A more frequent case is that a free
library does the same job as widely used non-free libraries.  In this
case, there is little to gain by limiting the free library to free
software only, so we use the Lesser General Public License.

  In other cases, permission to use a particular library in non-free
programs enables a greater number of people to use a large body of
free software.  For example, permission to use the GNU C Library in
non-free programs enables many more people to use the whole GNU
operating system, as well as its variant, the GNU/Linux operating
system.

  Although the Lesser General Public License is Less protective of the
users' freedom, it does ensure that the user of a program that is
linked with the Library has the freedom and the wherewithal to run
that program using a modified version of the Library.

  The precise terms and conditions for copying, distribution and
modification follow.  Pay close attention to the difference between a
"work based on the library" and a "work that uses the library".  The
former contains code derived from the library, whereas the latter must
be combined with the library in order to run.

                  GNU LESSER GENERAL PUBLIC LICENSE
   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

  0. This License Agreement applies to any software library or other
program which contains a notice placed by the copyright holder or
other authorized party saying it may be distributed under the terms of
this Lesser General Public License (also called "this License").
Each licensee is addressed as "you".

  A "library" means a collection of software functions and/or data
prepared so as to be conveniently linked with application programs
(which use some of those functions and data) to form executables.

  The "Library", below, refers to any such software library or work
which has been distributed under these terms.  A "work based on the
Library" means either the Library or any derivative work under
copyright law: that is to say, a work containing the Library or a
portion of it, either verbatim or with modifications and/or translated
straightforwardly into another language.  (Hereinafter, translation is
included without limitation in the term "modification".)

  "Source code" for a work means the preferred form of the work for
making modifications to it.  For a library, complete source code means
all the source code for all modules it contains, plus any associated
interface definition files, plus the scripts used to control compilation
and installation of the library.

  Activities other than copying, distribution and modification are not
covered by this License; they are outside its scope.  The act of
running a program using the Library is not restricted, and output from
such a program is covered only if its contents constitute a work based
on the Library (independent of the use of the Library in a tool for
writing it).  Whether that is true depends on what the Library does
```

and what the program that uses the Library does.

  1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

  You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

  2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

    a) The modified work must itself be a software library.

    b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

    c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

    d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

    (For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

  3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

  Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

  This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

  4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

  If distribution of object code is made by offering access to copy

from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer’s own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user’s computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is

normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries,

so that distribution is permitted only in or among countries not thus
excluded. In such case, this License incorporates the limitation as if
written in the body of this License.

  13. The Free Software Foundation may publish revised and/or new
versions of the Lesser General Public License from time to time.
Such new versions will be similar in spirit to the present version,
but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library
specifies a version number of this License which applies to it and
"any later version", you have the option of following the terms and
conditions either of that version or of any later version published by
the Free Software Foundation. If the Library does not specify a
license version number, you may choose any version ever published by
the Free Software Foundation.

  14. If you wish to incorporate parts of the Library into other free
programs whose distribution conditions are incompatible with these,
write to the author to ask for permission. For software which is
copyrighted by the Free Software Foundation, write to the Free
Software Foundation; we sometimes make exceptions for this. Our
decision will be guided by the two goals of preserving the free status
of all derivatives of our free software and of promoting the sharing
and reuse of software generally.

                            NO WARRANTY

  15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO
WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW.
EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR
OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY
KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE
LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME
THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN
WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY
AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU
FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR
CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE
LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING
RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A
FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF
SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH
DAMAGES.

                     END OF TERMS AND CONDITIONS

           How to Apply These Terms to Your New Libraries

  If you develop a new library, and you want it to be of the greatest
possible use to the public, we recommend making it free software that
everyone can redistribute and change. You can do so by permitting
redistribution under these terms (or, alternatively, under the terms of the
ordinary General Public License).

  To apply these terms, attach the following notices to the library. It is
safest to attach them to the start of each source file to most effectively
convey the exclusion of warranty; and each file should have at least the
"copyright" line and a pointer to where the full notice is found.

    <one line to give the library's name and a brief idea of what it does.>
    Copyright (C) <year> <name of author>

    This library is free software; you can redistribute it and/or
    modify it under the terms of the GNU Lesser General Public
    License as published by the Free Software Foundation; either
    version 2.1 of the License, or (at your option) any later version.

    This library is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
    Lesser General Public License for more details.

    You should have received a copy of the GNU Lesser General Public
    License along with this library; if not, write to the Free Software
    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the library, if

```
necessary.  Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the
  library `Frob' (a library for tweaking knobs) written by James Random Hacker.

  <signature of Ty Coon>, 1 April 1990
  Ty Coon, President of Vice

That's all there is to it!

------------------------------------------------------------------------

The bundled SHA1 collision detection code is licensed under the MIT license:

MIT License

Copyright (c) 2017:
    Marc Stevens
    Cryptology Group
    Centrum Wiskunde & Informatica
    P.O. Box 94079, 1090 GB Amsterdam, Netherlands
    marc@marc-stevens.nl

    Dan Shumow
    Microsoft Research
    danshu@microsoft.com

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

------------------------------------------------------------------------

The bundled wildmatch code is licensed under the BSD license:

Copyright Rich Salz.
All rights reserved.

Redistribution and use in any form are permitted provided that the
following restrictions are are met:

1.  Source distributions must retain this entire copyright notice
    and comment.
2.  Binary distributions must include the acknowledgement ``This
    product includes software developed by Rich Salz" in the
    documentation or other materials provided with the
    distribution.  This must not be represented as an endorsement
   or promotion without specific prior written permission.
3.  The origin of this software must not be misrepresented, either
    by explicit claim or by omission.  Credits must appear in the
    source and documentation.
4.  Altered versions must be plainly marked as such in the source
    and documentation and must not be misrepresented as being the
    original software.

THIS SOFTWARE IS PROVIDED ``AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED
WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

------------------------------------------------------------------------

Portions of the OpenSSL headers are included under the OpenSSL license:

Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
All rights reserved.

This package is an SSL implementation written
by Eric Young (eay@cryptsoft.com).
The implementation was written so as to conform with Netscapes SSL.

This library is free for commercial and non-commercial use as long as
the following conditions are aheared to.  The following conditions
apply to all code found in this distribution, be it the RC4, RSA,
```

```
lhash, DES, etc., code; not just the SSL code.  The SSL documentation
included with this distribution is covered by the same copyright terms
except that the holder is Tim Hudson (tjh@cryptsoft.com).

Copyright remains Eric Young's, and as such any Copyright notices in
the code are not to be removed.
If this package is used in a product, Eric Young should be given attribution
as the author of the parts of the library used.
This can be in the form of a textual message at program startup or
in documentation (online or textual) provided with the package.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software
   must display the following acknowledgement:
   "This product includes cryptographic software written by
    Eric Young (eay@cryptsoft.com)"
   The word 'cryptographic' can be left out if the rouines from the library
   being used are not cryptographic related :-).
4. If you include any Windows specific code (or a derivative thereof) from
   the apps directory (application code) you must include an acknowledgement:
   "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.

The licence and distribution terms for any publically available version or
derivative of this code cannot be changed.  i.e. this code cannot simply be
copied and put under another distribution licence
[including the GNU Public Licence.]

====================================================================
Copyright (c) 1998-2007 The OpenSSL Project.  All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in
   the documentation and/or other materials provided with the
   distribution.

3. All advertising materials mentioning features or use of this
   software must display the following acknowledgment:
   "This product includes software developed by the OpenSSL Project
   for use in the OpenSSL Toolkit. (http://www.openssl.org/)"

4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
   endorse or promote products derived from this software without
   prior written permission. For written permission, please contact
   openssl-core@openssl.org.

5. Products derived from this software may not be called "OpenSSL"
   nor may "OpenSSL" appear in their names without prior written
   permission of the OpenSSL Project.

6. Redistributions of any form whatsoever must retain the following
   acknowledgment:
   "This product includes software developed by the OpenSSL Project
   for use in the OpenSSL Toolkit (http://www.openssl.org/)"

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS" AND ANY
EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
```

```
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
OF THE POSSIBILITY OF SUCH DAMAGE.
```

2.1.3 libssh2/libssh2

```
/* Copyright (c) 2004-2007 Sara Golemon <sarag@libssh2.org>
 * Copyright (c) 2005,2006 Mikhail Gusarov <dottedmag@dottedmag.net>
 * Copyright (c) 2006-2007 The Written Word, Inc.
 * Copyright (c) 2007 Eli Fant <elifantu@mail.ru>
 * Copyright (c) 2009-2021 Daniel Stenberg
 * Copyright (C) 2008, 2009 Simon Josefsson
 * Copyright (c) 2000 Markus Friedl
 * Copyright (c) 2015 Microsoft Corp.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms,
 * with or without modification, are permitted provided
 * that the following conditions are met:
 *
 *   Redistributions of source code must retain the above
 *   copyright notice, this list of conditions and the
 *   following disclaimer.
 *
 *   Redistributions in binary form must reproduce the above
 *   copyright notice, this list of conditions and the following
 *   disclaimer in the documentation and/or other materials
 *   provided with the distribution.
 *
 *   Neither the name of the copyright holder nor the names
 *   of any other contributors may be used to endorse or
 *   promote products derived from this software without
 *   specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
 * CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
 * OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
 * BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
 * USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
 * OF SUCH DAMAGE.
 */
```

2.1.4 OpenSSL

```
  LICENSE ISSUES
  ==============

  The OpenSSL toolkit stays under a double license, i.e. both the conditions of
  the OpenSSL License and the original SSLeay license apply to the toolkit.
  See below for the actual license texts.

  OpenSSL License
  ---------------

/* ====================================================================
 * Copyright (c) 1998-2019 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
```

```
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

 Original SSLeay License
 -----------------------

/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
```

```
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
```

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

GdMUT 29
GdMUT.Components 29

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

AnimatedSprite2D
- Interactable . . . 67

Area2D
- Door . . . 60

CanvasLayer
- Hud . . . 64

CharacterBody2D
- BaseEnemy . . . 31
  - Boss1 . . . 50
- Player . . . 86

Damage . . . 57

Label
- BloodVial . . . 48

Node
- NavigationManager . . . 81
- PlayerStats . . . 101
- StorageManager . . . 121

Node2D
- Checkpoint . . . 55
- LevelManager . . . 70
- MainMenu . . . 72
- Spike . . . 114
- SpikeDynamic . . . 116

ParallaxLayer
- MainMenuBackground . . . 80

TextureProgressBar
- HealthBar . . . 62
- StaminaBar . . . 119

Chapter 5

Class Index

5.1 Class List

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

BaseEnemy
: Klasse für einen einfachen Gegner . . . 31

BloodVial
: Klasse für die Interaktion zum heilen . . . 48

Boss1
: Klasse für einen stärkeren Boss-Gegner, der von BaseEnemy erbt . . . 50

Checkpoint . . . 55

Damage
: Repräsentiert den Schaden, der von Charakteren oder Gegnern verursacht wird. Beinhaltet physischen Schaden, wahren Schaden und den Rückstoßeffekt . . . 57

Door
: Klasse für die Tür . . . 60

HealthBar
: Klasse für die Gesundheitsleiste des Spielers. Synchronisiert die Anzeige der HealthBar mit den Lebenspunkten des Spielers . . . 62

Hud
: Klasse für das PauseMenu . . . 64

Interactable
: Klasse für Interaktion . . . 67

LevelManager
: Klasse für den LevelManager Diese Klasse verwaltet den Levelwechsel und die Spielerpositionierung . . . 70

MainMenu
: Klasse für das MainMenu . . . 72

MainMenuBackground
: Klasse für die MainMenuBackground-Animation . . . 80

NavigationManager
: Der NavigationManager ist für das Laden von Leveln und das Spawnen des Spielers verantwortlich. Der NavigationManager ist ein Singleton, der in der Haupt-Szene platziert wird und von anderen Skripten verwendet wird, um Level zu laden und den Spieler zu spawnen . . . 81

Player
: Klasse für den Spielercharakter. Verwaltet Bewegung, Sprünge, Angriffe und Animationen . . . 86

PlayerStats
: Klasse für die Spielerstats . . . 101

Spike
: Klasse für die Spikes . . . 114

SpikeDynamic
Klasse für die beweglichen Spikes 116
StaminaBar
Klasse für die Ausdauerleiste des Spielers. Synchronisiert die Anzeige der StaminaBar mit der Ausdauer des Spielers 119
StorageManager
Klasse für das Speichern und Laden von Daten 121

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/TestClass.cs 159
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/addons/GDMUT/Dock.cs . . . 127
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/addons/GDMUT/GDMUT.cs . . . 129
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/addons/GDMUT/MethodResult.cs . . . 130
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/addons/GDMUT/Result.cs . . . 131
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/addons/GDMUT/TestFunction.cs . . . 131
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/addons/GDMUT/TestLoader.cs . . . 131
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/addons/GDMUT/TestResult.cs . . . 133
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/BaseEnemy.cs 134
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/BloodVial.cs 138
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Boss1.cs 139
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Checkpoint.cs 140
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Damage.cs 141
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Door.cs 141
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/HealthBar.cs 142
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Hud.cs 143
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Interactable.cs 143
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/LevelManager.cs 144

C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/MainMenu.cs 145
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/MainMenuBackground.cs 147
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/NavigationManager.cs 148
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Player.cs 149
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/PlayerStats.cs 153
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Spike.cs 155
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/SpikeDynamic.cs 156
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/StaminaBar.cs 157
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/StorageManager.cs 158

Chapter 7

Namespace Documentation

7.1 GdMUT Namespace Reference

Namespaces

- namespace Components

Classes

- class **TestClass**

  This is a test class for GDMUT. This is purely for demonstration. If you added this into your project, feel free to delete it =).

7.2 GdMUT.Components Namespace Reference

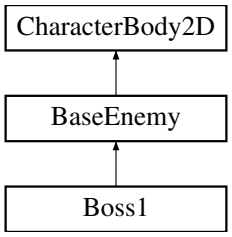
Chapter 8

Class Documentation

8.1 BaseEnemy Class Reference

Klasse für einen einfachen Gegner.

Inheritance diagram for BaseEnemy:



Public Member Functions

- override void _Ready ()

  Initialisierung der Referenzen. Findet die relevanten Knoten in der Szene und weist sie zu.
- override void _Process (double DeltaTime)

  Physikalische Prozesse werden in jedem Frame ausgeführt. Berechnet Gravitation und Bewegung.
- void OnDetectionBodyEntered (Node2D body)

  Detektiert den Spieler wenn er den Erkennungsbereich betritt.
- void OnPursuingRadiusBodyExited (Node2D body)

  Detektiert wenn der Spieler den Verfolgungsbereich verlässt.
- void OnHitboxAreaEntered (Area2D area)

  Detektiert wenn ein Objekt die Hitbox des Gegners betritt. (z.B.: Schwert des Spielers)
- void OnSwordHitBoxBodyEntered (Node2D body)

  Detektiert ob der Spieler in Schlagreichweite ist.
- bool IsDead ()

  Gibt boolean Dead zurück.
- void Respawn ()

  Wird aufgerufen wenn der Gegner respawnt.

Protected Member Functions

- virtual void UpdateAnimation ()

  Aktualisiert die Animationen des Gegners.

Protected Attributes

- float Damage = 20f
- bool Dead = false
- bool Respawnable = true
- float MaxHealthPoints = 100f
- float Armor = 20f
- float MaxStamina = 1f
- float Speed = 10
- int SinAmount = 10
- double ReturnToStartAfter = 5
- float CurrentHealthPoints
- float CurrentStamina
- double ReturnToStart
- bool Pursuing = false
- Node2D CurrentTarget = null
- Vector2 TargetPosition = Vector2.Inf
- Vector2 StartPosition
- bool StartRotation = false
- bool AlreadyHit = false
- AnimatedSprite2D Sprite
- CollisionPolygon2D CollisionPolygon
- Area2D SwordHitbox
- CollisionShape2D MainCollision
- RayCast2D FrontCollisionRayCast
- RayCast2D LineOfSight
- RayCast2D LeftFallProtection
- RayCast2D RightFallProtection
- TextureProgressBar HealthBar
- Player Player

Properties

- uint Id = 0 `[get, set]`

Private Types

- enum State { IDLE , WALK , ATTACK , TAKE_HIT }

Private Member Functions

- void HandleMovement (double DeltaTime)

  Verarbeitet die Bewegung des Gegners.
- void TakeDamage (Damage DMG)

  Verarbeitet zugefügten Schaden.
- void CheckPlayerHit ()

  Überprüft ob der Spieler sich, während eines Angriffes in Reichweite befindet und fügt diesem dann gegebenenfalls Schaden zu.
- void Die ()

  Wird aufgerufen wenn der Gegner stirbt.
- bool CheckLineOfSight (Node2D body)

  Überprüft die direkte Sichtlinie zu einem Objekt.
- void FlipRotation ()

  Spiegelt die Orientierung aller zu dem Gegner gehörender Nodes.
- void SetRotation (bool Rotation)

  Setzt Orientierung aller zu dem Gegner gehörender Nodes.
- bool IsCloseTo (float Value1, float Value2, float Delta)

  Überprüft, ob zwei Werte in einer Delta-Umgebung zueinander liegen.

Private Attributes

- State AnimationState = State.IDLE

8.1.1 Detailed Description

Klasse für einen einfachen Gegner.

Definition at line 7 of file BaseEnemy.cs.

8.1.2 Member Enumeration Documentation

8.1.2.1 State

```
enum BaseEnemy.State  [private]
```

Enumerator

IDLE	
WALK	
ATTACK	
TAKE_HIT	

Definition at line 10 of file BaseEnemy.cs.

```
00010                       {
00011         IDLE, WALK, ATTACK, TAKE_HIT
00012     }
```

8.1.3 Member Function Documentation

8.1.3.1 _Process()

```
override void BaseEnemy._Process (
            double DeltaTime)  [inline]
```

Physikalische Prozesse werden in jedem Frame ausgeführt. Berechnet Gravitation und Bewegung.

Parameters

DeltaTime	Zeit seit dem letzten Frame.

Definition at line 91 of file BaseEnemy.cs.

```
00092     {
00093         HandleMovement(DeltaTime);
00094         if(CurrentStamina < MaxStamina){
00095             CurrentStamina += (float) DeltaTime;
00096             Velocity = Velocity * 0.8f;
00097         }
00098         if (!IsOnFloor() && !Dead) {
00099             Velocity += GetGravity() * (float)DeltaTime;
00100         }
00101         UpdateAnimation();
00102         MoveAndSlide();
00103         CheckPlayerHit();
00104     }
```

References CheckPlayerHit(), CurrentStamina, Dead, HandleMovement(), MaxStamina, and UpdateAnimation().

8.1.3.2 _Ready()

```
override void BaseEnemy._Ready ()  [inline]
```

Initialisierung der Referenzen. Findet die relevanten Knoten in der Szene und weist sie zu.

Definition at line 64 of file BaseEnemy.cs.

```
00065     {
00066         Sprite = GetNode<AnimatedSprite2D>("AnimatedSprite2D");
00067         CollisionPolygon = GetNode<CollisionPolygon2D>("detection/CollisionPolygon2D");
00068         SwordHitbox = GetNode<Area2D>("AnimatedSprite2D/SwordHitBox");
00069         MainCollision = GetNode<CollisionShape2D>("MainCollision");
00070         FrontCollisionRayCast = GetNode<RayCast2D>("FrontCollisionRayCast");
00071         LineOfSight = GetNode<RayCast2D>("LineOfSight");
00072         LeftFallProtection = GetNode<RayCast2D>("LeftFallProtection");
00073         RightFallProtection = GetNode<RayCast2D>("RightFallProtection");
00074         HealthBar = GetNode<TextureProgressBar>("HealthBar");
00075         Player = GetNode<Player>("../../Player");
00076 
00077         CurrentHealthPoints = MaxHealthPoints;
00078         CurrentStamina = MaxStamina;
00079         ReturnToStart = ReturnToStartAfter;
00080         StartPosition = Position;
00081         StartRotation = Sprite.FlipH;
00082 
00083         HealthBar.Value = 100f* CurrentHealthPoints/MaxHealthPoints;
00084     }
```

References CollisionPolygon, CurrentHealthPoints, CurrentStamina, FrontCollisionRayCast, LeftFallProtection, LineOfSight, MainCollision, MaxHealthPoints, MaxStamina, ReturnToStart, ReturnToStartAfter, RightFallProtection, Sprite, StartPosition, StartRotation, and SwordHitbox.

8.1.3.3 CheckLineOfSight()

```
bool BaseEnemy.CheckLineOfSight (
            Node2D body)  [inline], [private]
```

Überprüft die direkte Sichtlinie zu einem Objekt.

Parameters

body	Objekt das überprüft werden soll.

Returns

bool Ergebnis der Abfrage.

Definition at line 334 of file BaseEnemy.cs.

```
00334                                                   {
00335         Vector2 offset = Vector2.Zero;
00336         offset.Y = -14;
00337         LineOfSight.TargetPosition = body.Position + offset - (Position + LineOfSight.Position);
00338         if(LineOfSight.IsColliding()){
00339             return LineOfSight.GetCollider() == body;
00340         }
00341         return true;
00342     }
```

References LineOfSight.

Referenced by OnDetectionBodyEntered().

8.1.3.4 CheckPlayerHit()

```
void BaseEnemy.CheckPlayerHit ()  [inline], [private]
```

Überprüft ob der Spieler sich, während eines Angriffes in Reichweite befindet und fügt diesem dann gegebenenfalls Schaden zu.

Definition at line 274 of file BaseEnemy.cs.

```
00274                                   {
00275         if(Dead) return;
00276         if(Sprite.Animation != "attack"){
00277             AlreadyHit = false;
00278             if(Sprite.Animation == "take_hit" || CurrentStamina < MaxStamina) return;
00279             Godot.Collections.Array<Node2D> Bodies = SwordHitbox.GetOverlappingBodies();
00280             foreach(Node2D Body in Bodies){
00281                 if(Body == Player){
00282                     Sprite.Play("attack");
00283                 }
00284             }
00285             return;
00286         }
00287         if(AlreadyHit) return;
00288         if(Sprite.Frame >= 6){
00289             CurrentStamina = 0;
00290             Godot.Collections.Array<Node2D> Bodies = SwordHitbox.GetOverlappingBodies();
00291             foreach(Node2D Body in Bodies){
00292                 if(Body == Player){
00293                     Player.TakeDamage(new Damage(Damage, 0f, Vector2.Zero, this));
00294                     AlreadyHit = true;
00295                     break;
00296                 }
00297             }
00298         }
00299
00300     }
```

References AlreadyHit, CurrentStamina, Damage, Dead, MaxStamina, Sprite, SwordHitbox, and Player.TakeDamage().

Referenced by _Process().

8.1.3.5 Die()

```
void BaseEnemy.Die ()  [inline], [private]
```

Wird aufgerufen wenn der Gegner stirbt.

Definition at line 305 of file BaseEnemy.cs.

```
00305                     {
00306         Dead = true;
00307         Velocity = Vector2.Zero;
00308         MainCollision.SetDeferred(CollisionShape2D.PropertyName.Disabled, true);
00309 
00310         Sprite.Play("death");
00311         HealthBar.SetVisible(false);
00312         Player.SetSinAmount(PlayerStats.Instance.GetSinAmount() + SinAmount);
00313 
00314     }
```

References Dead, PlayerStats.GetSinAmount(), PlayerStats.Instance, MainCollision, Player.SetSinAmount(), SinAmount, and Sprite.

Referenced by TakeDamage().

8.1.3.6 FlipRotation()

```
void BaseEnemy.FlipRotation ()  [inline], [private]
```

Spiegelt die Orientierung aller zu dem Gegner gehörender Nodes.

Definition at line 347 of file BaseEnemy.cs.

```
00347                         {
00348         Sprite.FlipH = !Sprite.FlipH;
00349         CollisionPolygon.RotationDegrees = Math.Abs(CollisionPolygon.RotationDegrees -180);
00350         SwordHitbox.RotationDegrees = Math.Abs(SwordHitbox.RotationDegrees -180);
00351         FrontCollisionRayCast.RotationDegrees = Math.Abs(FrontCollisionRayCast.RotationDegrees - 180);
00352     }
```

References CollisionPolygon, FrontCollisionRayCast, Sprite, and SwordHitbox.

Referenced by HandleMovement().

8.1.3.7 HandleMovement()

```
void BaseEnemy.HandleMovement (
             double DeltaTime)  [inline], [private]
```

Verarbeitet die Bewegung des Gegners.

Parameters

DeltaTime	Zeit seit dem letzten Frame.

Definition at line 150 of file BaseEnemy.cs.

```
00150                                                   {
00151         if(Dead) return;
00152         if((Sprite.Animation == "take_hit" || Sprite.Animation == "attack") && Sprite.IsPlaying()){
00153             Velocity = Vector2.Zero;
00154             return;
00155         }
00156         if(Pursuing){
00157             AnimationState = State.WALK;
00158             TargetPosition = CurrentTarget.Position;
00159             if(IsCloseTo(Position.X, TargetPosition.X, 0.1f)){
00160                 AnimationState = State.IDLE;
00161                 Velocity = Vector2.Zero;
00162                 return;
00163             }
00164             ReturnToStart = ReturnToStartAfter;
00165         } else if(ReturnToStart >= 0){
00166             AnimationState = State.IDLE;
00167             ReturnToStart -= DeltaTime;
00168             TargetPosition = Vector2.Inf;
00169         } else if(!IsCloseTo(Position.X, StartPosition.X, 0.1f)){
00170             AnimationState = State.WALK;
00171             TargetPosition = StartPosition;
00172         }
00173 
00174         if(TargetPosition != Vector2.Inf){
00175 
00176             if(IsCloseTo(Position.X, TargetPosition.X, 0.1f)){
00177                 AnimationState = State.IDLE;
00178                 Velocity = Vector2.Zero;
00179                 if(TargetPosition == StartPosition && Sprite.FlipH != StartRotation){
00180                     FlipRotation();
00181                 }
00182                 TargetPosition = Vector2.Inf;
00183                 return;
00184             }
00185 
00186             if(TargetPosition.X > Position.X){
00187                 SetRotation(true);
00188                 if(!FrontCollisionRayCast.IsColliding()){
00189                     Vector2 velocity = Vector2.Zero;
00190                     velocity.X = Speed;
00191                     Velocity = velocity;
00192                 }
00193             } else {
00194                 SetRotation(false);
00195                 if(!FrontCollisionRayCast.IsColliding()){
00196                     Vector2 velocity = Vector2.Zero;
00197                     velocity.X = -Speed;
00198                     Velocity = velocity;
00199                 }
00200             }
00201 
00202             if((!RightFallProtection.IsColliding() && !Sprite.FlipH) ||
      (!LeftFallProtection.IsColliding() && Sprite.FlipH)){
00203                 Velocity = Vector2.Zero;
00204             }
00205 
00206         } else {
00207             Velocity = Vector2.Zero;
00208             AnimationState = State.IDLE;
00209         }
00210     }
```

References AnimationState, CurrentTarget, Dead, FlipRotation(), FrontCollisionRayCast, IsCloseTo(), LeftFallProtection, Pursuing, ReturnToStart, ReturnToStartAfter, RightFallProtection, SetRotation(), Speed, Sprite, StartPosition, StartRotation, and TargetPosition.

Referenced by _Process().

8.1.3.8 IsCloseTo()

```
bool BaseEnemy.IsCloseTo (
            float Value1,
```

```
float Value2,
float Delta)  [inline], [private]
```

Überprüft, ob zwei Werte in einer Delta-Umgebung zueinander liegen.

Parameters

float	Wert1
float	Wert2
float	Delta

Returns

bool Ergebnis

Definition at line 378 of file BaseEnemy.cs.

```
00378                                                                   {
00379         return Value1 <= (Value2 + Delta) && Value1 >= (Value2 - Delta);
00380     }
```

Referenced by HandleMovement().

8.1.3.9 IsDead()

```
bool BaseEnemy.IsDead ()  [inline]
```

Gibt boolean Dead zurück.

Returns

bool ob Gegner tot ist.

Definition at line 266 of file BaseEnemy.cs.

```
00266                       {
00267         return Dead;
00268     }
```

References Dead.

Referenced by Boss1.ReviveEnemies().

8.1.3.10 OnDetectionBodyEntered()

```
void BaseEnemy.OnDetectionBodyEntered (
             Node2D body)  [inline]
```

Detektiert den Spieler wenn er den Erkennungsbereich betritt.

Parameters

body	Objekt das den Bereich betritt.

Definition at line 110 of file BaseEnemy.cs.

```
00110                                                   {
00111         if(CheckLineOfSight(body)){
00112             Pursuing = true;
00113             CurrentTarget = body;
00114         }
00115     }
```

References CheckLineOfSight(), CurrentTarget, and Pursuing.

8.1.3.11 OnHitboxAreaEntered()

```
void BaseEnemy.OnHitboxAreaEntered (
             Area2D area)  [inline]
```

Detektiert wenn ein Objekt die Hitbox des Gegners betritt. (z.B.: Schwert des Spielers)

Parameters

area	Objekt das den Bereich betritt.

Definition at line 132 of file BaseEnemy.cs.

```
00132                                                {
00133         Player Player1 = (Player) area.GetParent().GetParent();
00134         TakeDamage(Player1.GetDamage());
00135     }
```

References Player.GetDamage(), Player, and TakeDamage().

8.1.3.12 OnPursuingRadiusBodyExited()

```
void BaseEnemy.OnPursuingRadiusBodyExited (
             Node2D body)  [inline]
```

Detektiert wenn der Spieler den Verfolgungsbereich verlässt.

Parameters

body	Objekt das den Bereich verlässt.

Definition at line 121 of file BaseEnemy.cs.

```
00121                                                    {
00122         if(body == CurrentTarget){
00123             Pursuing = false;
00124             CurrentTarget = null;
00125         }
00126     }
```

References CurrentTarget, and Pursuing.

8.1.3.13 OnSwordHitBoxBodyEntered()

```
void BaseEnemy.OnSwordHitBoxBodyEntered (
             Node2D body)  [inline]
```

Detektiert ob der Spieler in Schlagreichweite ist.

Parameters

body	Objekt das den Bereich betritt.

Definition at line 141 of file BaseEnemy.cs.

```
00141                                                    {
00142         if(Dead) return;
00143         Sprite.Play("attack");
00144     }
```

References Dead, and Sprite.

8.1.3.14 Respawn()

```
void BaseEnemy.Respawn ()  [inline]
```

Wird aufgerufen wenn der Gegner respawnt.

Definition at line 319 of file BaseEnemy.cs.

```
00320     {
00321         Dead = false;
00322         CurrentHealthPoints = MaxHealthPoints;
00323         HealthBar.Value = 100f * CurrentHealthPoints / MaxHealthPoints;
00324         MainCollision.SetDeferred(CollisionShape2D.PropertyName.Disabled, false);
00325         HealthBar.SetVisible(true);
00326         Sprite.Play("idle");
00327     }
```

References CurrentHealthPoints, Dead, MainCollision, MaxHealthPoints, and Sprite.

Referenced by Boss1.ReviveEnemies().

8.1.3.15 SetRotation()

```
void BaseEnemy.SetRotation (
            bool Rotation)  [inline], [private]
```

Setzt Orientierung aller zu dem Gegner gehörender Nodes.

Parameters

Rotation	Die neue Orientierung.

Definition at line 358 of file BaseEnemy.cs.

```
00358                                           {
00359         Sprite.FlipH = Rotation ^ StartRotation; // XOR mit StartRotation
00360         if(Rotation){
00361             CollisionPolygon.RotationDegrees = 180;
00362             SwordHitbox.RotationDegrees = 180;
00363             FrontCollisionRayCast.RotationDegrees = 180;
00364         } else {
00365             CollisionPolygon.RotationDegrees = 0;
00366             SwordHitbox.RotationDegrees = 0;
00367             FrontCollisionRayCast.RotationDegrees = 0;
00368         }
00369     }
```

References StartRotation.

Referenced by HandleMovement().

8.1.3.16 TakeDamage()

```
void BaseEnemy.TakeDamage (
            Damage DMG)  [inline], [private]
```

Verarbeitet zugefügten Schaden.

Parameters

DMG	Schaden der zugefügt wird.

Definition at line 245 of file BaseEnemy.cs.

```
00245                                           {
00246         if(Dead) {
00247             return;
00248         }
00249         CurrentHealthPoints -= DMG.GetPhysicalDMG() * (1 - Armor / 100.0f) + DMG.GetTrueDMG();
00250         Position += DMG.GetPushAmount();
00251         if(CurrentHealthPoints <= 0){
00252             Die();
00253         } else {
00254             Sprite.Play("take_hit");
00255             if(DMG.GetSource() == Player){
00256                 Pursuing = true;
00257                 CurrentTarget = Player;
00258             }
00259         }
00260     }
```

References Armor, CurrentHealthPoints, CurrentTarget, Dead, Die(), Damage.GetPhysicalDMG(), Damage.GetPushAmount(), Damage.GetSource(), Damage.GetTrueDMG(), Player, Pursuing, and Sprite.

Referenced by OnHitboxAreaEntered().

8.1.3.17 UpdateAnimation()

```
virtual void BaseEnemy.UpdateAnimation ()  [inline], [protected], [virtual]
```

Aktualisiert die Animationen des Gegners.

Definition at line 216 of file BaseEnemy.cs.

```
00216                                           {
00217         if(Dead) return;
00218         if(!((Sprite.Animation == "take_hit" || Sprite.Animation == "attack") && Sprite.IsPlaying())){
00219             switch(AnimationState){
00220                 case State.IDLE:
00221                     Sprite.Play("idle");
00222                     break;
00223
00224                 case State.WALK:
00225                     Sprite.Play("walk");
00226                     break;
00227
00228                 case State.ATTACK:
00229                     Sprite.Play("attack");
00230                     break;
00231
00232                 case State.TAKE_HIT:
00233                     Sprite.Play("take_hit");
00234                     break;
00235             }
00236         }
00237         HealthBar.Value = 100f* CurrentHealthPoints/MaxHealthPoints;
00238
00239     }
```

References AnimationState, CurrentHealthPoints, Dead, MaxHealthPoints, and Sprite.

Referenced by _Process().

8.1.4 Member Data Documentation

8.1.4.1 AlreadyHit

```
bool BaseEnemy.AlreadyHit = false  [protected]
```

Definition at line 46 of file BaseEnemy.cs.

Referenced by CheckPlayerHit().

8.1.4.2 AnimationState

```
State BaseEnemy.AnimationState = State.IDLE  [private]
```

Definition at line 45 of file BaseEnemy.cs.

Referenced by HandleMovement(), and UpdateAnimation().

8.1.4.3 Armor

```
float BaseEnemy.Armor = 20f  [protected]
```

Definition at line 24 of file BaseEnemy.cs.

Referenced by Boss1._Process(), Boss1._Ready(), and TakeDamage().

8.1.4.4 CollisionPolygon

```
CollisionPolygon2D BaseEnemy.CollisionPolygon  [protected]
```

Definition at line 50 of file BaseEnemy.cs.

Referenced by _Ready(), and FlipRotation().

8.1.4.5 CurrentHealthPoints

```
float BaseEnemy.CurrentHealthPoints  [protected]
```

Definition at line 37 of file BaseEnemy.cs.

Referenced by Boss1._Process(), _Ready(), Boss1._Ready(), Boss1.HandleRegeneration(), Respawn(), TakeDamage(), and UpdateAnimation().

8.1.4.6 CurrentStamina

```
float BaseEnemy.CurrentStamina  [protected]
```

Definition at line 38 of file BaseEnemy.cs.

Referenced by _Process(), _Ready(), and CheckPlayerHit().

8.1.4.7 CurrentTarget

```
Node2D BaseEnemy.CurrentTarget = null  [protected]
```

Definition at line 41 of file BaseEnemy.cs.

Referenced by HandleMovement(), OnDetectionBodyEntered(), OnPursuingRadiusBodyExited(), and TakeDamage().

8.1.4.8 Damage

```
float BaseEnemy.Damage = 20f  [protected]
```

Definition at line 16 of file BaseEnemy.cs.

Referenced by CheckPlayerHit().

8.1.4.9 Dead

```
bool BaseEnemy.Dead = false  [protected]
```

Definition at line 18 of file BaseEnemy.cs.

Referenced by _Process(), CheckPlayerHit(), Die(), HandleMovement(), IsDead(), OnSwordHitBoxBodyEntered(), Respawn(), TakeDamage(), and UpdateAnimation().

8.1.4.10 FrontCollisionRayCast

```
RayCast2D BaseEnemy.FrontCollisionRayCast  [protected]
```

Definition at line 53 of file BaseEnemy.cs.

Referenced by _Ready(), FlipRotation(), and HandleMovement().

8.1.4.11 HealthBar

```
TextureProgressBar BaseEnemy.HealthBar  [protected]
```

Definition at line 57 of file BaseEnemy.cs.

8.1.4.12 LeftFallProtection

```
RayCast2D BaseEnemy.LeftFallProtection  [protected]
```

Definition at line 55 of file BaseEnemy.cs.

Referenced by _Ready(), and HandleMovement().

8.1.4.13 LineOfSight

```
RayCast2D BaseEnemy.LineOfSight  [protected]
```

Definition at line 54 of file BaseEnemy.cs.

Referenced by _Ready(), and CheckLineOfSight().

8.1.4.14 MainCollision

```
CollisionShape2D BaseEnemy.MainCollision  [protected]
```

Definition at line 52 of file BaseEnemy.cs.

Referenced by _Ready(), Die(), and Respawn().

8.1.4.15 MaxHealthPoints

```
float BaseEnemy.MaxHealthPoints = 100f  [protected]
```

Definition at line 22 of file BaseEnemy.cs.

Referenced by Boss1._Process(), _Ready(), Boss1._Ready(), Boss1.HandleRegeneration(), Respawn(), and UpdateAnimation().

8.1.4.16 MaxStamina

```
float BaseEnemy.MaxStamina = 1f  [protected]
```

Definition at line 26 of file BaseEnemy.cs.

Referenced by _Process(), _Ready(), and CheckPlayerHit().

8.1.4.17 Player

```
Player BaseEnemy.Player  [protected]
```

Definition at line 58 of file BaseEnemy.cs.

Referenced by OnHitboxAreaEntered(), and TakeDamage().

8.1.4.18 Pursuing

```
bool BaseEnemy.Pursuing = false  [protected]
```

Definition at line 40 of file BaseEnemy.cs.

Referenced by HandleMovement(), OnDetectionBodyEntered(), OnPursuingRadiusBodyExited(), and TakeDamage().

8.1.4.19 Respawnable

```
bool BaseEnemy.Respawnable = true  [protected]
```

Definition at line 20 of file BaseEnemy.cs.

8.1.4.20 ReturnToStart

```
double BaseEnemy.ReturnToStart  [protected]
```

Definition at line 39 of file BaseEnemy.cs.

Referenced by _Ready(), and HandleMovement().

8.1.4.21 ReturnToStartAfter

```
double BaseEnemy.ReturnToStartAfter = 5  [protected]
```

Definition at line 32 of file BaseEnemy.cs.

Referenced by _Ready(), and HandleMovement().

8.1.4.22 RightFallProtection

```
RayCast2D BaseEnemy.RightFallProtection  [protected]
```

Definition at line 56 of file BaseEnemy.cs.

Referenced by _Ready(), and HandleMovement().

8.1.4.23 SinAmount

```
int BaseEnemy.SinAmount = 10  [protected]
```

Definition at line 30 of file BaseEnemy.cs.

Referenced by Boss1._Ready(), and Die().

8.1.4.24 Speed

```
float BaseEnemy.Speed = 10  [protected]
```

Definition at line 28 of file BaseEnemy.cs.

Referenced by Boss1._Ready(), and HandleMovement().

8.1.4.25 Sprite

```
AnimatedSprite2D BaseEnemy.Sprite  [protected]
```

Definition at line 49 of file BaseEnemy.cs.

Referenced by _Ready(), CheckPlayerHit(), Die(), FlipRotation(), HandleMovement(), OnSwordHitBoxBodyEntered(), Respawn(), Boss1.StartGlowing(), TakeDamage(), and UpdateAnimation().

8.1.4.26 StartPosition

```
Vector2 BaseEnemy.StartPosition  [protected]
```

Definition at line 43 of file BaseEnemy.cs.

Referenced by _Ready(), and HandleMovement().

8.1.4.27 StartRotation

```
bool BaseEnemy.StartRotation = false  [protected]
```

Definition at line 44 of file BaseEnemy.cs.

Referenced by _Ready(), HandleMovement(), and SetRotation().

8.1.4.28 SwordHitbox

```
Area2D BaseEnemy.SwordHitbox  [protected]
```

Definition at line 51 of file BaseEnemy.cs.

Referenced by _Ready(), CheckPlayerHit(), and FlipRotation().

8.1.4.29 TargetPosition

```
Vector2 BaseEnemy.TargetPosition = Vector2.Inf  [protected]
```

Definition at line 42 of file BaseEnemy.cs.

Referenced by HandleMovement().

8.1.5 Property Documentation

8.1.5.1 Id

```
uint BaseEnemy.Id = 0  [get], [set]
```

Definition at line 34 of file BaseEnemy.cs.

```
00034 { get; set;} = 0;
```

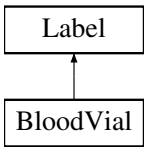
The documentation for this class was generated from the following file:

- C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/BaseEnemy.cs

8.2 BloodVial Class Reference

Klasse für die Interaktion zum heilen.

Inheritance diagram for BloodVial:



Public Member Functions

- override void _Ready ()

  Initialisierung der Referenzen. Findet die relevanten Knoten in der Szene und weist sie zu.
- void UseBloodVial ()

  Versucht ein Bloodvial zu verwenden um den Spieler zu Heilen.
- void ResetUses ()

  Setzt die Anzahl der Bloodvials auf das Maximum.
- void AddMaxUses (int Amount)

  Verbessert die Maximale Anzahl an Bloodvials um die angegebene Anzahl.
- void LevelHealAmount ()

  Verbessert den HealAMount eines Bloodvials um 25.

8.2.1 Detailed Description

Klasse für die Interaktion zum heilen.

Definition at line 8 of file BloodVial.cs.

8.2.2 Member Function Documentation

8.2.2.1 _Ready()

```
override void BloodVial._Ready ()  [inline]
```

Initialisierung der Referenzen. Findet die relevanten Knoten in der Szene und weist sie zu.

Definition at line 14 of file BloodVial.cs.

```
00014                                   {
00015         Text = PlayerStats.Instance.GetBVCurrentUses() + "";
00016     }
```

References PlayerStats.GetBVCurrentUses(), and PlayerStats.Instance.

8.2.2.2 AddMaxUses()

```
void BloodVial.AddMaxUses (
            int Amount)  [inline]
```

Verbessert die Maximale Anzahl an Bloodvials um die angegebene Anzahl.

Parameters

int	Amount, um die MaxUses erhöht wird.

Definition at line 40 of file BloodVial.cs.

```
00040                                       {
00041         PlayerStats.Instance.SetBVMaxUses(PlayerStats.Instance.GetBVMaxUses() + Amount);
00042         ResetUses();
00043     }
```

References PlayerStats.GetBVMaxUses(), PlayerStats.Instance, ResetUses(), and PlayerStats.SetBVMaxUses().

8.2.2.3 LevelHealAmount()

```
void BloodVial.LevelHealAmount ()  [inline]
```

Verbessert den HealAMount eines Bloodvials um 25.

Definition at line 48 of file BloodVial.cs.

```
00048                                   {
00049         PlayerStats.Instance.SetBVHealAmount(PlayerStats.Instance.GetBVHealAmount() + 25);
00050     }
```

References PlayerStats.GetBVHealAmount(), PlayerStats.Instance, and PlayerStats.SetBVHealAmount().

8.2.2.4 ResetUses()

```
void BloodVial.ResetUses ()  [inline]
```

Setzt die Anzahl der Bloodvials auf das Maximum.

Definition at line 31 of file BloodVial.cs.

```
00031                               {
00032         PlayerStats.Instance.SetBVCurrentUses(PlayerStats.Instance.GetBVMaxUses());
00033         Text = PlayerStats.Instance.GetBVCurrentUses() + "";
00034     }
```

References PlayerStats.GetBVCurrentUses(), PlayerStats.GetBVMaxUses(), PlayerStats.Instance, and PlayerStats.SetBVCurrentUse

Referenced by AddMaxUses(), Checkpoint.OnPlayerBodyEntered(), and Player.Respawn().

8.2.2.5 UseBloodVial()

```
void BloodVial.UseBloodVial ()  [inline]
```

Versucht ein Bloodvial zu verwenden um den Spieler zu Heilen.

Definition at line 21 of file BloodVial.cs.

```
00021                                 {
00022         if(PlayerStats.Instance.GetBVCurrentUses() <= 0) return;
00023         PlayerStats.Instance.SetBVCurrentUses(PlayerStats.Instance.GetBVCurrentUses() - 1);
00024         Text = PlayerStats.Instance.GetBVCurrentUses() + "";
00025         PlayerStats.Instance.SetCurrentHealth(PlayerStats.Instance.GetCurrentHealth() +
      PlayerStats.Instance.GetBVHealAmount());
00026     }
```

References PlayerStats.GetBVCurrentUses(), PlayerStats.GetBVHealAmount(), PlayerStats.GetCurrentHealth(), PlayerStats.Instance, PlayerStats.SetBVCurrentUses(), and PlayerStats.SetCurrentHealth().

Referenced by Player._PhysicsProcess().

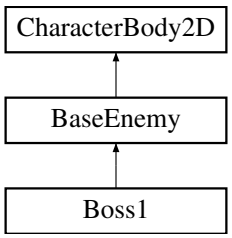
The documentation for this class was generated from the following file:

- C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/BloodVial.cs

8.3 Boss1 Class Reference

Klasse für einen stärkeren Boss-Gegner, der von BaseEnemy erbt.

Inheritance diagram for Boss1:



Public Member Functions

- override void _Ready ()

  Überschreibt die _Ready-Methode von BaseEnemy.
- override void _Process (double DeltaTime)

  Überschreibt die _Process-Methode von BaseEnemy.

Public Member Functions inherited from BaseEnemy

- override void _Ready ()

  Initialisierung der Referenzen. Findet die relevanten Knoten in der Szene und weist sie zu.
- override void _Process (double DeltaTime)

  Physikalische Prozesse werden in jedem Frame ausgeführt. Berechnet Gravitation und Bewegung.
- void OnDetectionBodyEntered (Node2D body)

  Detektiert den Spieler wenn er den Erkennungsbereich betritt.
- void OnPursuingRadiusBodyExited (Node2D body)

  Detektiert wenn der Spieler den Verfolgungsbereich verlässt.
- void OnHitboxAreaEntered (Area2D area)

  Detektiert wenn ein Objekt die Hitbox des Gegners betritt. (z.B.: Schwert des Spielers)
- void OnSwordHitBoxBodyEntered (Node2D body)

  Detektiert ob der Spieler in Schlagreichweite ist.
- bool IsDead ()

  Gibt boolean Dead zurück.
- void Respawn ()

  Wird aufgerufen wenn der Gegner respawnt.

Private Member Functions

- void HandleRegeneration (double DeltaTime)

  Regeneriert die Gesundheit des Bosses, wenn er keinen Schaden nimmt.
- void StartGlowing ()

  Startet einen Leuchteffekt, wenn der Boss Schaden nimmt.
- void ShowPopupMessage (string Message)

  Zeigt eine Popup-Nachricht an.
- void ReviveEnemies ()

  Lässt alle toten Feinde im Raum des Bosses wiederbeleben.

Private Attributes

- bool EnemiesRevived = false
- float RegenCooldown = 5.0f
- float RegenTimer = 0.0f
- float RegenAmount = 10.0f

Additional Inherited Members

Protected Member Functions inherited from BaseEnemy

- virtual void UpdateAnimation ()

  Aktualisiert die Animationen des Gegners.

Protected Attributes inherited from BaseEnemy

- float Damage = 20f
- bool Dead = false
- bool Respawnable = true
- float MaxHealthPoints = 100f
- float Armor = 20f
- float MaxStamina = 1f
- float Speed = 10
- int SinAmount = 10
- double ReturnToStartAfter = 5
- float CurrentHealthPoints
- float CurrentStamina
- double ReturnToStart
- bool Pursuing = false
- Node2D CurrentTarget = null
- Vector2 TargetPosition = Vector2.Inf
- Vector2 StartPosition
- bool StartRotation = false
- bool AlreadyHit = false
- AnimatedSprite2D Sprite
- CollisionPolygon2D CollisionPolygon
- Area2D SwordHitbox
- CollisionShape2D MainCollision
- RayCast2D FrontCollisionRayCast
- RayCast2D LineOfSight
- RayCast2D LeftFallProtection
- RayCast2D RightFallProtection
- TextureProgressBar HealthBar
- Player Player

Properties inherited from BaseEnemy

- uint Id = 0 `[get, set]`

8.3.1 Detailed Description

Klasse für einen stärkeren Boss-Gegner, der von BaseEnemy erbt.

Definition at line 7 of file Boss1.cs.

8.3.2 Member Function Documentation

8.3.2.1 _Process()

```
override void Boss1._Process (
            double DeltaTime)  [inline]
```

Überschreibt die _Process-Methode von BaseEnemy.

Parameters

DeltaTime	Die Zeit, die seit dem letzten Frame vergangen ist

Definition at line 36 of file Boss1.cs.

```
00036                                                   {
00037         base._Process(DeltaTime);
00038
00039         if (CurrentHealthPoints <= MaxHealthPoints / 2 && !EnemiesRevived){
00040             StartGlowing();
00041             ReviveEnemies();
00042             EnemiesRevived = true;
00043             Armor = 60f; // Rüstung erhöhen
00044         }
00045
00046         HandleRegeneration(DeltaTime);
00047     }
```

References BaseEnemy.Armor, BaseEnemy.CurrentHealthPoints, EnemiesRevived, HandleRegeneration(), BaseEnemy.MaxHealthPoints, ReviveEnemies(), and StartGlowing().

8.3.2.2 _Ready()

```
override void Boss1._Ready ()  [inline]
```

Überschreibt die _Ready-Methode von BaseEnemy.

Definition at line 18 of file Boss1.cs.

```
00018                                   {
00019
00020         MaxHealthPoints = 400f;
00021         Damage = 50f;
00022         Armor = 30f;
00023         Speed = 10f;
00024         SinAmount = 100; // Bonuspunkte für Spieler beim Besiegen des Bosses
00025
00026         base._Ready();
00027
00028         CurrentHealthPoints = MaxHealthPoints;
00029         HealthBar.Value = 100f * CurrentHealthPoints / MaxHealthPoints;
00030     }
```

References BaseEnemy.Armor, BaseEnemy.CurrentHealthPoints, BaseEnemy.MaxHealthPoints, BaseEnemy.SinAmount, and BaseEnemy.Speed.

8.3.2.3 HandleRegeneration()

```
void Boss1.HandleRegeneration (
            double DeltaTime)  [inline], [private]
```

Regeneriert die Gesundheit des Bosses, wenn er keinen Schaden nimmt.

Parameters

DeltaTime	Die Zeit, die seit dem letzten Frame vergangen ist

Definition at line 53 of file Boss1.cs.

```
00053                                                {
00054         if (CurrentHealthPoints < MaxHealthPoints){
00055             RegenTimer += (float)DeltaTime;
00056 
00057             if (RegenTimer >= RegenCooldown){
00058                 CurrentHealthPoints = Math.Min(CurrentHealthPoints + RegenAmount, MaxHealthPoints);
00059                 HealthBar.Value = 100f * CurrentHealthPoints / MaxHealthPoints;
00060                 RegenTimer = 0.0f; // Timer zurücksetzen
00061             }
00062         }
00063     }
```

References BaseEnemy.CurrentHealthPoints, BaseEnemy.MaxHealthPoints, RegenAmount, RegenCooldown, and RegenTimer.

Referenced by _Process().

8.3.2.4 ReviveEnemies()

```
void Boss1.ReviveEnemies ()  [inline], [private]
```

Lässt alle toten Feinde im Raum des Bosses wiederbeleben.

Definition at line 108 of file Boss1.cs.

```
00109     {
00110         // Hole den Elternknoten (bossRoom)
00111         Node BossRoom = GetParent();
00112 
00113         // Iteriere durch alle Kinder von bossRoom
00114         foreach (Node Child in BossRoom.GetChildren()){
00115             if (Child is BaseEnemy BaseEnemy && BaseEnemy.IsDead()){
00116                 BaseEnemy.Respawn();
00117             }
00118         }
00119     }
```

References BaseEnemy.IsDead(), and BaseEnemy.Respawn().

Referenced by _Process().

8.3.2.5 ShowPopupMessage()

```
void Boss1.ShowPopupMessage (
             string Message)  [inline], [private]
```

Zeigt eine Popup-Nachricht an.

Parameters

Message	Die Nachricht, die angezeigt werden soll

Definition at line 80 of file Boss1.cs.

```
00080                                                {
00081         Label popup = new Label();
00082         popup.Text = Message;
00083         popup.AddThemeColorOverride("font_color", new Color(1, 0, 0)); // Rot
00084         popup.Modulate = new Color(1, 1, 1, 0); // Start transparent
00085         popup.AutowrapMode = TextServer.AutowrapMode.Word;
00086         popup.SizeFlagsHorizontal = (Control.SizeFlags)(int)Control.SizeFlags.ExpandFill;
00087         popup.SizeFlagsVertical = (Control.SizeFlags)(int)Control.SizeFlags.ShrinkCenter;
00088         popup.HorizontalAlignment = HorizontalAlignment.Center;
00089         popup.VerticalAlignment = VerticalAlignment.Center;
00090
00091
00092         Vector2 bossGlobalPosition = GetGlobalTransformWithCanvas().Origin;
00093         popup.GlobalPosition = bossGlobalPosition + new Vector2(0, -100);
00094
00095         CanvasLayer canvas = new CanvasLayer();
00096         AddChild(canvas);
00097         canvas.AddChild(popup);
00098
00099         var tween = CreateTween();
00100         tween.TweenProperty(popup, "modulate:a", 1, 0.5f).From(0); // Einblenden
00101         tween.TweenProperty(popup, "modulate:a", 0, 0.5f).From(1).SetDelay(1.0f); // Ausblenden nach 1
      Sekunde
00102         tween.Connect("finished", new Callable(popup, "queue_free"));
00103     }
```

Referenced by StartGlowing().

8.3.2.6 StartGlowing()

```
void Boss1.StartGlowing ()  [inline], [private]
```

Startet einen Leuchteffekt, wenn der Boss Schaden nimmt.

Definition at line 68 of file Boss1.cs.

```
00068                                {
00069         // Ändere die Modulationsfarbe des Sprites, um ein Leuchten zu simulieren
00070         if (Sprite != null){
00071             ShowPopupMessage("AHHHH!!!");
00072             Sprite.Modulate = new Color(1.0f, 0.84f, 0.0f, 1.0f); // Ein goldliche Leuchteffekt
00073         }
00074     }
```

References ShowPopupMessage(), and BaseEnemy.Sprite.

Referenced by _Process().

8.3.3 Member Data Documentation

8.3.3.1 EnemiesRevived

```
bool Boss1.EnemiesRevived = false  [private]
```

Definition at line 9 of file Boss1.cs.

Referenced by _Process().

8.3.3.2 RegenAmount

```
float Boss1.RegenAmount = 10.0f  [private]
```

Definition at line 12 of file Boss1.cs.

Referenced by HandleRegeneration().

8.3.3.3 RegenCooldown

```
float Boss1.RegenCooldown = 5.0f  [private]
```

Definition at line 10 of file Boss1.cs.

Referenced by HandleRegeneration().

8.3.3.4 RegenTimer

```
float Boss1.RegenTimer = 0.0f  [private]
```

Definition at line 11 of file Boss1.cs.

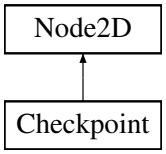
Referenced by HandleRegeneration().

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

- C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Boss1.cs

8.4 Checkpoint Class Reference

Inheritance diagram for Checkpoint:



Public Member Functions

- override void _Ready ()

Private Member Functions

- void OnPlayerBodyEntered (Node body)

Private Attributes

- Player Player

8.4.1 Detailed Description

Definition at line 4 of file Checkpoint.cs.

8.4.2 Member Function Documentation

8.4.2.1 _Ready()

```
override void Checkpoint._Ready ()  [inline]
```

Definition at line 13 of file Checkpoint.cs.

```
00014     {
00015         // Zugriff auf Player Node
00016         Player = GetNode<Player>("../Player");
00017     }
```

8.4.2.2 OnPlayerBodyEntered()

```
void Checkpoint.OnPlayerBodyEntered (
            Node body)  [inline], [private]
```

Prüfen ob der Körper, der den Checkpoint betritt, ein Player ist Wenn ja, dann wird der Checkpoint als Spawnpoint gesetzt

Definition at line 23 of file Checkpoint.cs.

```
00024     {
00025
00031         if (body is Player Player)
00032         {
00033             // Setzen des Spawnpoints
00034             PlayerStats PlayerStats = GetNode<PlayerStats>("/root/PlayerStats");
00035             PlayerStats.Instance.SetSpawnPoint(this.GlobalPosition);
00036             Player.MaxHeal();
00037             PlayerStats.Instance.SetStamina(PlayerStats.Instance.GetMaxStamina());
00038             Player.GetBloodVials().ResetUses();
00039             GD.Print("Spawnpoint des Players gesetzt auf: ", this.GlobalPosition);
00040
00041             PlayerStats.SetRespawnLevelTag(GetParent().Name);
00042             GD.Print("RespawnLevelTag des Players gesetzt auf: ", GetParent().Name);
00043             GD.Print(PlayerStats.Instance.GetRespawnLevelTag());
00044         }
00045
00046     }
```

References Player.GetBloodVials(), PlayerStats.GetMaxStamina(), PlayerStats.GetRespawnLevelTag(), PlayerStats.Instance, Player.MaxHeal(), BloodVial.ResetUses(), PlayerStats.SetRespawnLevelTag(), PlayerStats.SetSpawnPoint(), and PlayerStats.SetStamina().

8.4.3 Member Data Documentation

8.4.3.1 Player

```
Player Checkpoint.Player  [private]
```

Definition at line 8 of file Checkpoint.cs.

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

- C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Checkpoint.cs

8.5 Damage Class Reference

Repräsentiert den Schaden, der von Charakteren oder Gegnern verursacht wird. Beinhaltet physischen Schaden, wahren Schaden und den Rückstoßeffekt.

Public Member Functions

- Damage (float PhysicalDMG, float TrueDMG, Vector2 PushAmount, Node2D Source)

  Konstruktor für die Damage-Klasse.
- float GetPhysicalDMG ()

  Gibt den physischen Schaden zurück.
- float GetTrueDMG ()

  Gibt den wahren Schaden zurück.
- Vector2 GetPushAmount ()

  Gibt den Rückstoßvektor zurück.
- Node2D GetSource ()

  Gibt die Ursache zurück.

Private Attributes

- float PhysicalDMG
- float TrueDMG
- Vector2 PushAmount
- Node2D Source

8.5.1 Detailed Description

Repräsentiert den Schaden, der von Charakteren oder Gegnern verursacht wird. Beinhaltet physischen Schaden, wahren Schaden und den Rückstoßeffekt.

Definition at line 7 of file Damage.cs.

8.5.2 Constructor & Destructor Documentation

8.5.2.1 Damage()

```
Damage.Damage (
            float PhysicalDMG,
            float TrueDMG,
            Vector2 PushAmount,
            Node2D Source)  [inline]
```

Konstruktor für die Damage-Klasse.

Parameters

PhysicalDMG	Der physische Schaden.
TrueDMG	Der wahre Schaden.
PushAmount	Der Rückstoßvektor.

Definition at line 20 of file Damage.cs.

```
00020                                                                    {
00021         this.PhysicalDMG = PhysicalDMG;
00022         this.TrueDMG = TrueDMG;
00023         this.PushAmount = PushAmount;
00024         this.Source = Source;
00025     }
```

References PhysicalDMG, PushAmount, Source, and TrueDMG.

8.5.3 Member Function Documentation

8.5.3.1 GetPhysicalDMG()

```
float Damage.GetPhysicalDMG ()  [inline]
```

Gibt den physischen Schaden zurück.

Returns

Der physische Schaden.

Definition at line 31 of file Damage.cs.

```
00031                                   {
00032         return PhysicalDMG;
00033     }
```

References PhysicalDMG.

Referenced by BaseEnemy.TakeDamage(), and Player.TakeDamage().

8.5.3.2 GetPushAmount()

```
Vector2 Damage.GetPushAmount ()  [inline]
```

Gibt den Rückstoßvektor zurück.

Returns

Der Rückstoßvektor.

Definition at line 47 of file Damage.cs.

```
00047                                    {
00048         return PushAmount;
00049     }
```

References PushAmount.

Referenced by BaseEnemy.TakeDamage(), and Player.TakeDamage().

8.5.3.3 GetSource()

```
Node2D Damage.GetSource ()  [inline]
```

Gibt die Ursache zurück.

Returns

Die Ursache.

Definition at line 55 of file Damage.cs.

```
00055                                  {
00056         return Source;
00057     }
```

References Source.

Referenced by BaseEnemy.TakeDamage().

8.5.3.4 GetTrueDMG()

```
float Damage.GetTrueDMG ()  [inline]
```

Gibt den wahren Schaden zurück.

Returns

Der wahre Schaden.

Definition at line 39 of file Damage.cs.

```
00039                                {
00040         return TrueDMG;
00041     }
```

References TrueDMG.

Referenced by BaseEnemy.TakeDamage(), and Player.TakeDamage().

8.5.4 Member Data Documentation

8.5.4.1 PhysicalDMG

```
float Damage.PhysicalDMG  [private]
```

Definition at line 9 of file Damage.cs.

Referenced by Damage(), and GetPhysicalDMG().

8.5.4.2 PushAmount

```
Vector2 Damage.PushAmount  [private]
```

Definition at line 11 of file Damage.cs.

Referenced by Damage(), and GetPushAmount().

8.5.4.3 Source

```
Node2D Damage.Source  [private]
```

Definition at line 12 of file Damage.cs.

Referenced by Damage(), and GetSource().

8.5.4.4 TrueDMG

```
float Damage.TrueDMG  [private]
```

Definition at line 10 of file Damage.cs.

Referenced by Damage(), and GetTrueDMG().

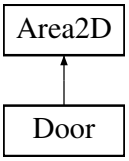
The documentation for this class was generated from the following file:

- C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Damage.cs

8.6 Door Class Reference

Klasse für die Tür.

Inheritance diagram for Door:



Public Member Functions

- override void _Ready ()
  Initialisierung der Node Spawn.

Public Attributes

- Node Spawn

Properties

- string DestinationLevelTag `[get, set]`
- string DestinationDoorTag `[get, set]`
- string SpawnDirection = "up" `[get, set]`

Private Member Functions

- void OnPlayerBodyEntered (Node body)
  Diese Funktion wird aufgerufen, wenn der Player die Tür betritt.

8.6.1 Detailed Description

Klasse für die Tür.

Die Klasse ist für den Wechsel zwischen den Levels zuständig.

Definition at line 8 of file Door.cs.

8.6.2 Member Function Documentation

8.6.2.1 _Ready()

```
override void Door._Ready ()  [inline]
```

Initialisierung der Node Spawn.

Definition at line 26 of file Door.cs.

```
00027     {
00028         Spawn = GetNode("Spawn");
00029     }
```

References Spawn.

8.6.2.2 OnPlayerBodyEntered()

```
void Door.OnPlayerBodyEntered (
            Node body)  [inline], [private]
```

Diese Funktion wird aufgerufen, wenn der Player die Tür betritt.

Parameters

body	Der Körper, der die Tür betritt

Definition at line 36 of file Door.cs.

```
00037     {
00038         if (body is Player player)
00039         {
00040             var NavigationManager = GetNode<NavigationManager>("/root/NavigationManager");
00041             NavigationManager.GoToLevel(DestinationLevelTag, DestinationDoorTag);
00042         }
00043     }
```

References DestinationDoorTag, DestinationLevelTag, and NavigationManager.GoToLevel().

8.6.3 Member Data Documentation

8.6.3.1 Spawn

```
Node Door.Spawn
```

Definition at line 10 of file Door.cs.

Referenced by _Ready().

8.6.4 Property Documentation

8.6.4.1 DestinationDoorTag

```
string Door.DestinationDoorTag  [get], [set]
```

Definition at line 16 of file Door.cs.
```
00016 { get; set; }
```

Referenced by OnPlayerBodyEntered().

8.6.4.2 DestinationLevelTag

```
string Door.DestinationLevelTag  [get], [set]
```

Definition at line 13 of file Door.cs.
```
00013 { get; set; }
```

Referenced by OnPlayerBodyEntered().

8.6.4.3 SpawnDirection

```
string Door.SpawnDirection = "up"  [get], [set]
```

Definition at line 19 of file Door.cs.
```
00019 { get; set; } = "up";
```

Referenced by LevelManager.OnLevelSpawn().

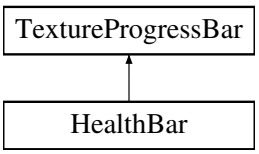
The documentation for this class was generated from the following file:

- C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Door.cs

8.7 HealthBar Class Reference

Klasse für die Gesundheitsleiste des Spielers. Synchronisiert die Anzeige der HealthBar mit den Lebenspunkten des Spielers.

Inheritance diagram for HealthBar:



Public Member Functions

- override void _Ready ()

  Initialisiert die HealthBar und verbindet sie mit den Lebenspunkten des Spielers. Lädt den Spieler-Knoten und setzt die maximale und aktuelle Gesundheit in der HealthBar.
- override void _Process (double DeltaTime)

  Aktualisiert die HealthBar in jedem Frame. Synchronisiert die Anzeige der aktuellen Lebenspunkte mit den Werten des Spielers.

8.7.1 Detailed Description

Klasse für die Gesundheitsleiste des Spielers. Synchronisiert die Anzeige der HealthBar mit den Lebenspunkten des Spielers.

Definition at line 7 of file HealthBar.cs.

8.7.2 Member Function Documentation

8.7.2.1 _Process()

```
override void HealthBar._Process (
            double DeltaTime)  [inline]
```

Aktualisiert die HealthBar in jedem Frame. Synchronisiert die Anzeige der aktuellen Lebenspunkte mit den Werten des Spielers.

Parameters

delta	Zeit seit dem letzten Frame (wird nicht direkt genutzt).

Definition at line 24 of file HealthBar.cs.

```
00024                                           {
00025         // Aktualisiere den Wert der HealthBar basierend auf der aktuellen Gesundheit des Spielers
00026         Value = PlayerStats.Instance.GetCurrentHealth();
00027     }
```

References PlayerStats.GetCurrentHealth(), and PlayerStats.Instance.

8.7.2.2 _Ready()

```
override void HealthBar._Ready ()  [inline]
```

Initialisiert die HealthBar und verbindet sie mit den Lebenspunkten des Spielers. Lädt den Spieler-Knoten und setzt die maximale und aktuelle Gesundheit in der HealthBar.

Definition at line 13 of file HealthBar.cs.

```
00013                                 {
00014         // Setze die maximale Gesundheit der HealthBar basierend auf der Spieler-MaxHealth
00015         MaxValue = PlayerStats.Instance.GetMaxHealthPoints();
00016         Value = PlayerStats.Instance.GetCurrentHealth();
00017     }
```

References PlayerStats.GetCurrentHealth(), PlayerStats.GetMaxHealthPoints(), and PlayerStats.Instance.

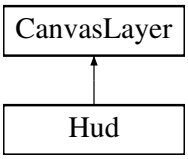
The documentation for this class was generated from the following file:

- C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/HealthBar.cs

8.8 Hud Class Reference

Klasse für das PauseMenu.

Inheritance diagram for Hud:



Public Member Functions

- override void _Ready ()
  Initialisierung der Referenzen. Findet die relevanten Knoten in der Szene und weist sie zu.
- override void _Process (double DeltaTime)
  Methode wird in jedem Frame ausgeführt.
- void OnResumeButtonPressed ()
  Signal für den Resume-Button.
- void OnSaveButtonPressed ()
  Signal für den Save-Button.
- void OnSaveMenuButtonPressed ()
  Signal für den SaveAndReturnToMenu-Button.
- void OnSaveQuitButtonPressed ()
  Signal für den SaveAndQuit-Button.

Private Member Functions

- void TogglePause ()
  Toggled die Pause Funktion.

Private Attributes

- AnimationPlayer AnimationPlayer
- CenterContainer Buttons
- bool Enabled

8.8.1 Detailed Description

Klasse für das PauseMenu.

Definition at line 8 of file Hud.cs.

8.8.2 Member Function Documentation

8.8.2.1 _Process()

```
override void Hud._Process (
            double DeltaTime)  [inline]
```

Methode wird in jedem Frame ausgeführt.

Parameters

DeltaTime	Zeit seit dem letzten Frame.

Definition at line 29 of file Hud.cs.

```
00029                                                   {
00030         if(Input.IsActionJustPressed("escape")){
00031             TogglePause();
00032         }
00033     }
```

References TogglePause().

8.8.2.2 _Ready()

```
override void Hud._Ready ()  [inline]
```

Initialisierung der Referenzen. Findet die relevanten Knoten in der Szene und weist sie zu.

Definition at line 19 of file Hud.cs.

```
00019                                {
00020         AnimationPlayer = GetNode<AnimationPlayer>("PauseMenu/AnimationPlayer");
00021         Buttons = GetNode<CenterContainer>("PauseMenu/Buttons");
00022         AnimationPlayer.Play("RESET");
00023     }
```

References AnimationPlayer, and Buttons.

8.8.2.3 OnResumeButtonPressed()

```
void Hud.OnResumeButtonPressed ()  [inline]
```

Signal für den Resume-Button.

Definition at line 53 of file Hud.cs.

```
00053                                        {
00054         TogglePause();
00055     }
```

References TogglePause().

8.8.2.4 OnSaveButtonPressed()

```
void Hud.OnSaveButtonPressed ()  [inline]
```

Signal für den Save-Button.

Definition at line 60 of file Hud.cs.

```
00060                                      {
00061         StorageManager.Instance.SaveAll(StorageManager.Instance.GetLastSaveId());
00062     }
```

References StorageManager.GetLastSaveId(), StorageManager.Instance, and StorageManager.SaveAll().

8.8.2.5 OnSaveMenuButtonPressed()

```
void Hud.OnSaveMenuButtonPressed ()  [inline]
```

Signal für den SaveAndReturnToMenu-Button.

Definition at line 67 of file Hud.cs.

```
00067                                       {
00068         StorageManager.Instance.SaveAll(StorageManager.Instance.GetLastSaveId());
00069         NavigationManager.Instance.GoToLevel("main_menu", null);
00070         PlayerStats.Instance.Reload();
00071         GetTree().Paused = false;
00072     }
```

References StorageManager.GetLastSaveId(), NavigationManager.GoToLevel(), NavigationManager.Instance, PlayerStats.Instance, StorageManager.Instance, PlayerStats.Reload(), and StorageManager.SaveAll().

8.8.2.6 OnSaveQuitButtonPressed()

```
void Hud.OnSaveQuitButtonPressed ()  [inline]
```

Signal für den SaveAndQuit-Button.

Definition at line 77 of file Hud.cs.

```
00077                                       {
00078         StorageManager.Instance.SaveAll(StorageManager.Instance.GetLastSaveId());
00079         GetTree().Quit();
00080     }
```

References StorageManager.GetLastSaveId(), StorageManager.Instance, and StorageManager.SaveAll().

8.8.2.7 TogglePause()

```
void Hud.TogglePause ()  [inline], [private]
```

Toggled die Pause Funktion.

Definition at line 38 of file Hud.cs.

```
00038                                   {
00039         Enabled = !Enabled;
00040         GetTree().Paused = Enabled;
00041         if(Enabled){
00042             AnimationPlayer.Play("Pause");
00043             Buttons.Visible = true;
00044         } else {
00045             AnimationPlayer.PlayBackwards("Pause");
00046             Buttons.Visible = false;
00047         }
00048     }
```

References AnimationPlayer, and Enabled.

Referenced by _Process(), and OnResumeButtonPressed().

8.8.3 Member Data Documentation

8.8.3.1 AnimationPlayer

```
AnimationPlayer Hud.AnimationPlayer  [private]
```

Definition at line 10 of file Hud.cs.

Referenced by _Ready(), and TogglePause().

8.8.3.2 Buttons

```
CenterContainer Hud.Buttons  [private]
```

Definition at line 11 of file Hud.cs.

Referenced by _Ready().

8.8.3.3 Enabled

```
bool Hud.Enabled  [private]
```

Definition at line 12 of file Hud.cs.

Referenced by TogglePause().

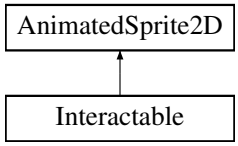
The documentation for this class was generated from the following file:

- C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Hud.cs

8.9 Interactable Class Reference

Klasse für Interaktion.

Inheritance diagram for Interactable:



Public Member Functions

- override void _Ready ()

  Initialisierung der Referenzen. Findet die relevanten Knoten in der Szene und weist sie zu.
- override void _Process (double DeltaTime)

  Testet, ob der Spieler mit der Node Interagiert und öffnet ein PopUp.
- void OnAreaBodyExited (Node2D Body)

  Detektiert, wenn der Spieler den Bereich verlässt und schließt das PopUp.

Properties

- String Text `[get, set]`

Private Attributes

- Player Player
- RichTextLabel TextLabel
- Control PopUp
- Area2D Area

8.9.1 Detailed Description

Klasse für Interaktion.

Definition at line 7 of file Interactable.cs.

8.9.2 Member Function Documentation

8.9.2.1 _Process()

```
override void Interactable._Process (
            double DeltaTime)  [inline]
```

Testet, ob der Spieler mit der Node Interagiert und öffnet ein PopUp.

Parameters

DeltaTime	Zeit zwischen den Frames.

Definition at line 32 of file Interactable.cs.

```
00032                                                {
00033         if(Input.IsActionJustPressed("interact")){
00034             Godot.Collections.Array<Node2D> Bodies = Area.GetOverlappingBodies();
00035             foreach(Node2D Body in Bodies){
00036                 if(Body == Player){
00037                     TextLabel.Clear();
00038                     TextLabel.AppendText(Text);
00039                     PopUp.Visible = true;
00040                     return;
00041                 }
00042             }
00043         }
00044     }
```

References Area, Text, and TextLabel.

8.9.2.2 _Ready()

```
override void Interactable._Ready ()  [inline]
```

Initialisierung der Referenzen. Findet die relevanten Knoten in der Szene und weist sie zu.

Definition at line 21 of file Interactable.cs.

```
00021                                {
00022         Player = GetNode<Player>("../Player");
00023         TextLabel = GetNode<RichTextLabel>("../HUD/PopUp/Text");
00024         PopUp = GetNode<Control>("../HUD/PopUp");
00025         Area = GetNode<Area2D>("Area2D");
00026     }
```

References Area, PopUp, and TextLabel.

8.9.2.3 OnAreaBodyExited()

```
void Interactable.OnAreaBodyExited (
            Node2D Body)  [inline]
```

Detektiert, wenn der Spieler den Bereich verlässt und schließt das PopUp.

Parameters

Node2D	die den Bereich verlässt.

Definition at line 50 of file Interactable.cs.

```
00050                                                   {
00051         if(Body == Player){
00052             PopUp.Visible = false;
00053             TextLabel.Clear();
00054         }
00055     }
```

References TextLabel.

8.9.3 Member Data Documentation

8.9.3.1 Area

```
Area2D Interactable.Area  [private]
```

Definition at line 12 of file Interactable.cs.

Referenced by _Process(), and _Ready().

8.9.3.2 Player

```
Player Interactable.Player  [private]
```

Definition at line 9 of file Interactable.cs.

8.9.3.3 PopUp

```
Control Interactable.PopUp  [private]
```

Definition at line 11 of file Interactable.cs.

Referenced by _Ready().

8.9.3.4 TextLabel

```
RichTextLabel Interactable.TextLabel  [private]
```

Definition at line 10 of file Interactable.cs.

Referenced by _Process(), _Ready(), and OnAreaBodyExited().

8.9.4 Property Documentation

8.9.4.1 Text

```
String Interactable.Text  [get], [set], [private]
```

Definition at line 15 of file Interactable.cs.

```
00015 { get; set;}
```

Referenced by _Process().

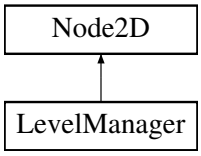
The documentation for this class was generated from the following file:

- C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Interactable.cs

8.10 LevelManager Class Reference

Klasse für den LevelManager Diese Klasse verwaltet den Levelwechsel und die Spielerpositionierung.

Inheritance diagram for LevelManager:



Public Member Functions

- override void _Ready ()

  Initialisierung der Referenzen. Findet die relevanten Knoten in der Szene und weist sie zu.

Private Member Functions

- void OnLevelSpawn (string DestinationTag)

  Wird aufgerufen, wenn ein neues Level geladen wird.

8.10.1 Detailed Description

Klasse für den LevelManager Diese Klasse verwaltet den Levelwechsel und die Spielerpositionierung.

Definition at line 7 of file LevelManager.cs.

8.10.2 Member Function Documentation

8.10.2.1 _Ready()

```
override void LevelManager._Ready ()  [inline]
```

Initialisierung der Referenzen. Findet die relevanten Knoten in der Szene und weist sie zu.

Wenn ein Spawn-Tag gesetzt ist, wird der Spieler an die entsprechende Tür gesetzt. Dies wird verwendet, um den Spieler an eine bestimmte Tür zu setzen, wenn er von einem anderen Level aus spawnt.

Definition at line 13 of file LevelManager.cs.

```
00014     {
00015         var NavigationManager = GetNode<NavigationManager>("/root/NavigationManager");
00016 
00021         if (NavigationManager.SpawnDoorTag != null)
00022         {
00023             OnLevelSpawn(NavigationManager.SpawnDoorTag);
00024         }
00025         else
00026         {
00027             NavigationManager.CallDeferred("TriggerPlayerSpawn", PlayerStats.Instance.GetPosition(),
      "");
00028         }
00029 
00030     }
```

References PlayerStats.GetPosition(), PlayerStats.Instance, OnLevelSpawn(), and NavigationManager.SpawnDoorTag.

8.10.2.2 OnLevelSpawn()

```
void LevelManager.OnLevelSpawn (
             string DestinationTag)  [inline], [private]
```

Wird aufgerufen, wenn ein neues Level geladen wird.

Parameters

DestinationTag	Das Tag der Tür, an der der Spieler spawnen soll.

Definition at line 36 of file LevelManager.cs.

```
00037     {
00038         var NavigationManager = GetNode<NavigationManager>("/root/NavigationManager");
00039         // Pfad zur Tür basierend auf dem Ziel-Tag erstellen
00040         string DoorPath = "Doors/Door_" + DestinationTag;
00041 
00042         Door door = GetNode<Door>(DoorPath);
00043 
00044         // TriggerPlayerSpawn nach deferred ausführen
00045         NavigationManager.CallDeferred("TriggerPlayerSpawn", door.GlobalPosition,
      door.SpawnDirection);
00046     }
```

References Door.SpawnDirection.

Referenced by _Ready().

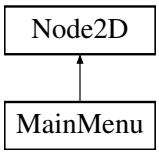
The documentation for this class was generated from the following file:

- C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/LevelManager.cs

8.11 MainMenu Class Reference

Klasse für das MainMenu.

Inheritance diagram for MainMenu:



Public Member Functions

- override void _Ready ()

  Initialisierung der Referenzen. Findet die relevanten Knoten in der Szene und weist sie zu.
- void OnContinueButtonPressed ()

  Signal für den Continue-Button.
- void OnQuitButtonPressed ()

  Signal für den Quit-Button.
- void OnNewGameButtonPressed ()

  Signal für den NewGame-Button.
- void OnLoadGameButtonPressed ()

  Signal für den LoadGame-Button.
- void OnBackButtonPressed ()

  Signal für den Back-Button.
- void OnSave1SelectPressed ()

  Signal für den Select1-Button.
- void OnSave1DeletePressed ()

  Signal für den Delete1-Button.
- void OnSave2SelectPressed ()

  Signal für den Select2-Button.
- void OnSave2DeletePressed ()

  Signal für den Delete2-Button.
- void OnSave3SelectPressed ()

  Signal für den Select3-Button.
- void OnSave3DeletePressed ()

  Signal für den Delete3-Button.
- void OnDeleteConfirmationCanceled ()

  Signal für den Delete-Abbruch.
- void OnDeleteConfirmationConfirmed ()

  Signal für die Delete-Bestätigung.
- void OnDeleteConfirmationCloseRequested ()

  Signal für das Schließen der Delete-Bestätigung.

Private Member Functions

- void Change ()

  Wechselt das Menu zwischen den verschiedenen States.

Private Attributes

- int MenuState = 0
- VBoxContainer Navigation
- MarginContainer SavesContainer
- Button ContinueButton
- Label InfoLabel
- Label[ ] SaveLabel = new Label[3]
- Button[ ] SelectButton = new Button[3]
- Button[ ] DeleteButton = new Button[3]
- ConfirmationDialog DeleteConfirmation
- int SaveToDelete = 0

8.11.1 Detailed Description

Klasse für das MainMenu.

Definition at line 7 of file MainMenu.cs.

8.11.2 Member Function Documentation

8.11.2.1 _Ready()

```
override void MainMenu._Ready ()  [inline]
```

Initialisierung der Referenzen. Findet die relevanten Knoten in der Szene und weist sie zu.

Definition at line 25 of file MainMenu.cs.

```
00025                               {
00026         Navigation = GetNode<VBoxContainer>("Control/Navigation");
00027         SavesContainer = GetNode<MarginContainer>("Control/Saves");
00028         ContinueButton = GetNode<Button>("Control/Navigation/ContinueButton");
00029         InfoLabel = GetNode<Label>("Control/Saves/VBoxContainer/Info");
00030 
00031         SaveLabel[0] = GetNode<Label>("Control/Saves/VBoxContainer/HBoxContainer/Save1/Label");
00032         SelectButton[0] = GetNode<Button>("Control/Saves/VBoxContainer/HBoxContainer/Save1/Select");
00033         DeleteButton[0] = GetNode<Button>("Control/Saves/VBoxContainer/HBoxContainer/Save1/Delete");
00034         SaveLabel[1] = GetNode<Label>("Control/Saves/VBoxContainer/HBoxContainer/Save2/Label");
00035         SelectButton[1] = GetNode<Button>("Control/Saves/VBoxContainer/HBoxContainer/Save2/Select");
00036         DeleteButton[1] = GetNode<Button>("Control/Saves/VBoxContainer/HBoxContainer/Save2/Delete");
00037         SaveLabel[2] = GetNode<Label>("Control/Saves/VBoxContainer/HBoxContainer/Save3/Label");
00038         SelectButton[2] = GetNode<Button>("Control/Saves/VBoxContainer/HBoxContainer/Save3/Select");
00039         DeleteButton[2] = GetNode<Button>("Control/Saves/VBoxContainer/HBoxContainer/Save3/Delete");
00040 
00041         DeleteConfirmation = GetNode<ConfirmationDialog>("DeleteConfirmation");
00042 
00043         if(StorageManager.Instance.GetLastSaveId() > -1){
00044             ContinueButton.Visible = true;
00045         }
00046     }
```

References ContinueButton, DeleteButton, DeleteConfirmation, StorageManager.GetLastSaveId(), InfoLabel, StorageManager.Instance, Navigation, SaveLabel, SavesContainer, and SelectButton.

8.11.2.2 Change()

```
void MainMenu.Change ()  [inline], [private]
```

Wechselt das Menu zwischen den verschiedenen States.

Definition at line 52 of file MainMenu.cs.

```
00052                           {
00053         if(MenuState == 0){
00054             SavesContainer.Visible = false;
00055             Navigation.Visible = true;
00056         } else {
00057             Navigation.Visible = false;
00058             SavesContainer.Visible = true;
00059 
00060             int Saves = StorageManager.Instance.GetSaves();
00061 
00062             if(MenuState == 1){
00063                 InfoLabel.Text = "Select empty save to start a new Game";
00064                 for(int i = 0; i < 3; i++){
00065                     if((Saves & (int) Math.Pow(2, i)) == (int) Math.Pow(2, i)){
00066                         SaveLabel[i].Text = "Save " + (i+1);
00067                         SelectButton[i].Disabled = true;
00068                         DeleteButton[i].Disabled = false;
00069                     } else {
00070                         SaveLabel[i].Text = "Save " + (i+1) + "\nEmpty";
00071                         SelectButton[i].Disabled = false;
00072                         DeleteButton[i].Disabled = true;
00073                     }
00074                 }
00075             } else {
00076                 InfoLabel.Text = "Select save to load Game";
00077                 for(int i = 0; i < 3; i++){
00078                     if((Saves & (int) Math.Pow(2, i)) == (int) Math.Pow(2, i)){
00079                         SaveLabel[i].Text = "Save " + (i+1);
00080                         SelectButton[i].Disabled = false;
00081                         DeleteButton[i].Disabled = false;
00082                     } else {
00083                         SaveLabel[i].Text = "Save " + (i+1) + "\nEmpty";
00084                         SelectButton[i].Disabled = true;
00085                         DeleteButton[i].Disabled = true;
00086                     }
00087                 }
00088             }
00089         }
00090     }
```

References DeleteButton, StorageManager.GetSaves(), StorageManager.Instance, MenuState, SaveLabel, and SelectButton.

Referenced by OnBackButtonPressed(), OnDeleteConfirmationCanceled(), OnDeleteConfirmationConfirmed(), OnLoadGameButtonPressed(), and OnNewGameButtonPressed().

8.11.2.3 OnBackButtonPressed()

```
void MainMenu.OnBackButtonPressed ()  [inline]
```

Signal für den Back-Button.

Definition at line 127 of file MainMenu.cs.

```
00127                                   {
00128         MenuState = 0;
00129         Change();
00130     }
```

References Change(), and MenuState.

8.11.2.4 OnContinueButtonPressed()

```
void MainMenu.OnContinueButtonPressed ()  [inline]
```

Signal für den Continue-Button.

Definition at line 95 of file MainMenu.cs.

```
00095                                       {
00096         StorageManager.Instance.LoadGameFile(StorageManager.Instance.GetLastSaveId());
00097         NavigationManager.Instance.GoToLevel(PlayerStats.Instance.GetCurrentLevelTag(), null);
00098     }
```

References PlayerStats.GetCurrentLevelTag(), StorageManager.GetLastSaveId(), NavigationManager.GoToLevel(), NavigationManager.Instance, PlayerStats.Instance, StorageManager.Instance, and StorageManager.LoadGameFile().

8.11.2.5 OnDeleteConfirmationCanceled()

```
void MainMenu.OnDeleteConfirmationCanceled ()  [inline]
```

Signal für den Delete-Abbruch.

Definition at line 198 of file MainMenu.cs.

```
00198                                           {
00199         SaveToDelete = 0;
00200         Change();
00201     }
```

References Change(), and SaveToDelete.

Referenced by OnDeleteConfirmationCloseRequested().

8.11.2.6 OnDeleteConfirmationCloseRequested()

```
void MainMenu.OnDeleteConfirmationCloseRequested ()  [inline]
```

Signal für das Schließen der Delete-Bestätigung.

Definition at line 214 of file MainMenu.cs.

```
00214                                                {
00215         OnDeleteConfirmationCanceled();
00216     }
```

References OnDeleteConfirmationCanceled().

8.11.2.7 OnDeleteConfirmationConfirmed()

```
void MainMenu.OnDeleteConfirmationConfirmed ()  [inline]
```

Signal für die Delete-Bestätigung.

Definition at line 206 of file MainMenu.cs.

```
00206                                           {
00207         StorageManager.Instance.SetSaves(StorageManager.Instance.GetSaves() ^ (int) Math.Pow(2,
      SaveToDelete - 1));
00208         Change();
00209     }
```

References Change(), StorageManager.GetSaves(), StorageManager.Instance, SaveToDelete, and StorageManager.SetSaves().

8.11.2.8 OnLoadGameButtonPressed()

```
void MainMenu.OnLoadGameButtonPressed ()  [inline]
```

Signal für den LoadGame-Button.

Definition at line 119 of file MainMenu.cs.

```
00119                                       {
00120         MenuState = 2;
00121         Change();
00122     }
```

References Change(), and MenuState.

8.11.2.9 OnNewGameButtonPressed()

```
void MainMenu.OnNewGameButtonPressed ()  [inline]
```

Signal für den NewGame-Button.

Definition at line 111 of file MainMenu.cs.

```
00111                                       {
00112         MenuState = 1;
00113         Change();
00114     }
```

References Change(), and MenuState.

8.11.2.10 OnQuitButtonPressed()

```
void MainMenu.OnQuitButtonPressed ()  [inline]
```

Signal für den Quit-Button.

Definition at line 103 of file MainMenu.cs.

```
00103                                     {
00104         StorageManager.Instance.SaveSettings();
00105         GetTree().Quit();
00106     }
```

References StorageManager.Instance, and StorageManager.SaveSettings().

8.11.2.11 OnSave1DeletePressed()

```
void MainMenu.OnSave1DeletePressed ()  [inline]
```

Signal für den Delete1-Button.

Definition at line 147 of file MainMenu.cs.

```
00147                                       {
00148         SaveToDelete = 1;
00149         DeleteConfirmation.SetText("Are you sure you want to DELETE Save " + SaveToDelete + "?");
00150         DeleteConfirmation.Show();
00151     }
```

References DeleteConfirmation, and SaveToDelete.

8.11.2.12 OnSave1SelectPressed()

```
void MainMenu.OnSave1SelectPressed ()  [inline]
```

Signal für den Select1-Button.

Definition at line 135 of file MainMenu.cs.

```
00135                                   {
00136         if(MenuState == 2){
00137             StorageManager.Instance.LoadGameFile(0);
00138         }
00139         NavigationManager.Instance.GoToLevel(PlayerStats.Instance.GetCurrentLevelTag(), null);
00140         StorageManager.Instance.SetSaves(StorageManager.Instance.GetSaves() | 1);
00141         StorageManager.Instance.SetLastSaveId(0);
00142     }
```

References PlayerStats.GetCurrentLevelTag(), StorageManager.GetSaves(), NavigationManager.GoToLevel(), NavigationManager.Instance, PlayerStats.Instance, StorageManager.Instance, StorageManager.LoadGameFile(), MenuState, StorageManager.SetLastSaveId(), and StorageManager.SetSaves().

8.11.2.13 OnSave2DeletePressed()

```
void MainMenu.OnSave2DeletePressed ()  [inline]
```

Signal für den Delete2-Button.

Definition at line 168 of file MainMenu.cs.

```
00168                                   {
00169         SaveToDelete = 2;
00170         DeleteConfirmation.SetText("Are you sure you want to DELETE Save " + SaveToDelete + "?");
00171         DeleteConfirmation.Show();
00172     }
```

References DeleteConfirmation, and SaveToDelete.

8.11.2.14 OnSave2SelectPressed()

```
void MainMenu.OnSave2SelectPressed ()  [inline]
```

Signal für den Select2-Button.

Definition at line 156 of file MainMenu.cs.

```
00156                                   {
00157         if(MenuState == 2){
00158             StorageManager.Instance.LoadGameFile(1);
00159         }
00160         NavigationManager.Instance.GoToLevel(PlayerStats.Instance.GetCurrentLevelTag(), null);
00161         StorageManager.Instance.SetSaves(StorageManager.Instance.GetSaves() | 2);
00162         StorageManager.Instance.SetLastSaveId(1);
00163     }
```

References PlayerStats.GetCurrentLevelTag(), StorageManager.GetSaves(), NavigationManager.GoToLevel(), NavigationManager.Instance, PlayerStats.Instance, StorageManager.Instance, StorageManager.LoadGameFile(), MenuState, StorageManager.SetLastSaveId(), and StorageManager.SetSaves().

8.11.2.15 OnSave3DeletePressed()

```
void MainMenu.OnSave3DeletePressed ()  [inline]
```

Signal für den Delete3-Button.

Definition at line 189 of file MainMenu.cs.

```
00189                                   {
00190         SaveToDelete = 3;
00191         DeleteConfirmation.SetText("Are you sure you want to DELETE Save " + SaveToDelete + "?");
00192         DeleteConfirmation.Show();
00193     }
```

References DeleteConfirmation, and SaveToDelete.

8.11.2.16 OnSave3SelectPressed()

```
void MainMenu.OnSave3SelectPressed ()  [inline]
```

Signal für den Select3-Button.

Definition at line 177 of file MainMenu.cs.

```
00177                                   {
00178         if(MenuState == 2){
00179             StorageManager.Instance.LoadGameFile(2);
00180         }
00181         NavigationManager.Instance.GoToLevel(PlayerStats.Instance.GetCurrentLevelTag(), null);
00182         StorageManager.Instance.SetSaves(StorageManager.Instance.GetSaves() | 4);
00183         StorageManager.Instance.SetLastSaveId(2);
00184     }
```

References PlayerStats.GetCurrentLevelTag(), StorageManager.GetSaves(), NavigationManager.GoToLevel(), NavigationManager.Instance, PlayerStats.Instance, StorageManager.Instance, StorageManager.LoadGameFile(), MenuState, StorageManager.SetLastSaveId(), and StorageManager.SetSaves().

8.11.3 Member Data Documentation

8.11.3.1 ContinueButton

```
Button MainMenu.ContinueButton  [private]
```

Definition at line 12 of file MainMenu.cs.

Referenced by _Ready().

8.11.3.2 DeleteButton

```
Button [] MainMenu.DeleteButton = new Button[3]  [private]
```

Definition at line 16 of file MainMenu.cs.

Referenced by _Ready(), and Change().

8.11.3.3 DeleteConfirmation

```
ConfirmationDialog MainMenu.DeleteConfirmation  [private]
```

Definition at line 17 of file MainMenu.cs.

Referenced by _Ready(), OnSave1DeletePressed(), OnSave2DeletePressed(), and OnSave3DeletePressed().

8.11.3.4 InfoLabel

```
Label MainMenu.InfoLabel  [private]
```

Definition at line 13 of file MainMenu.cs.

Referenced by _Ready().

8.11.3.5 MenuState

```
int MainMenu.MenuState = 0  [private]
```

Definition at line 9 of file MainMenu.cs.

Referenced by Change(), OnBackButtonPressed(), OnLoadGameButtonPressed(), OnNewGameButtonPressed(), OnSave1SelectPressed(), OnSave2SelectPressed(), and OnSave3SelectPressed().

8.11.3.6 Navigation

```
VBoxContainer MainMenu.Navigation  [private]
```

Definition at line 10 of file MainMenu.cs.

Referenced by _Ready().

8.11.3.7 SaveLabel

```
Label [] MainMenu.SaveLabel = new Label[3]  [private]
```

Definition at line 14 of file MainMenu.cs.

Referenced by _Ready(), and Change().

8.11.3.8 SavesContainer

```
MarginContainer MainMenu.SavesContainer  [private]
```

Definition at line 11 of file MainMenu.cs.

Referenced by _Ready().

8.11.3.9 SaveToDelete

```
int MainMenu.SaveToDelete = 0  [private]
```

Definition at line 18 of file MainMenu.cs.

Referenced by OnDeleteConfirmationCanceled(), OnDeleteConfirmationConfirmed(), OnSave1DeletePressed(), OnSave2DeletePressed(), and OnSave3DeletePressed().

8.11.3.10 SelectButton

```
Button [] MainMenu.SelectButton = new Button[3]  [private]
```

Definition at line 15 of file MainMenu.cs.

Referenced by _Ready(), and Change().

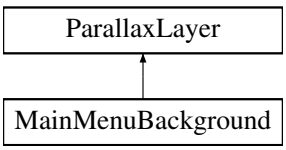
The documentation for this class was generated from the following file:

- C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/MainMenu.cs

8.12 MainMenuBackground Class Reference

Klasse für die MainMenuBackground-Animation.

Inheritance diagram for MainMenuBackground:



Public Member Functions

- override void _Process (double DeltaTime)
  Methode wird in jedem Frame ausgeführt.

Private Attributes

- float ScrollSpeed = -10f

8.12.1 Detailed Description

Klasse für die MainMenuBackground-Animation.

Definition at line 7 of file MainMenuBackground.cs.

8.12.2 Member Function Documentation

8.12.2.1 _Process()

```
override void MainMenuBackground._Process (
            double DeltaTime)  [inline]
```

Methode wird in jedem Frame ausgeführt.

Parameters

DeltaTime	Zeit seit dem letzten Frame.

Definition at line 16 of file MainMenuBackground.cs.

```
00016                                                   {
00017         float X = GetMotionOffset().X;
00018         X += ScrollSpeed * (float) DeltaTime;
00019         SetMotionOffset(new Vector2(X,0));
00020     }
```

References ScrollSpeed.

8.12.3 Member Data Documentation

8.12.3.1 ScrollSpeed

```
float MainMenuBackground.ScrollSpeed = -10f  [private]
```

Definition at line 10 of file MainMenuBackground.cs.

Referenced by _Process().

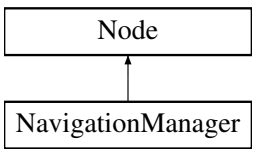
The documentation for this class was generated from the following file:

- C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/MainMenuBackground.cs

8.13 NavigationManager Class Reference

Der NavigationManager ist für das Laden von Leveln und das Spawnen des Spielers verantwortlich. Der NavigationManager ist ein Singleton, der in der Haupt-Szene platziert wird und von anderen Skripten verwendet wird, um Level zu laden und den Spieler zu spawnen.

Inheritance diagram for NavigationManager:



Public Member Functions

- delegate void OnTriggerPlayerSpawnEventHandler (Vector2 Position, string Direction)

  Das Signal, das ausgelöst wird, wenn der Spieler spawnen soll.
- override void _Ready ()

  Initialisiert den NavigationManager und setzt ihn als Singleton.
- void GoToLevel (string LevelTag, string DestinationTag)

  Lädt das angegebene Level und setzt das Ziel-Tag für den Spieler-Spawn.
- void TriggerPlayerSpawn (Vector2 Position, string Direction)

  Lädt das angegebene Level und setzt das Ziel-Tag für den Spieler-Spawn.

Properties

- static NavigationManager Instance `[get, private set]`
- string SpawnDoorTag `[get, private set]`

Private Member Functions

- void DeferredChangeScene (PackedScene SceneToLoad)

  Diese Methode wird aufgerufen, um die Szene zu wechseln.

Static Private Attributes

- static readonly PackedScene SceneMainMenu = (PackedScene)GD.Load("res://Scenes/main_menu.tscn")
- static readonly PackedScene SceneIntro = (PackedScene)GD.Load("res://Scenes/intro.tscn")
- static readonly PackedScene SceneLevel1 = (PackedScene)GD.Load("res://Scenes/level1.tscn")
- static readonly PackedScene SceneBoss = (PackedScene)GD.Load("res://Scenes/bossRoom.tscn")
- static readonly PackedScene SceneLevelOne = (PackedScene)GD.Load("res://Scenes/level_one.tscn")
- static readonly PackedScene SceneLevelTwo = (PackedScene)GD.Load("res://Scenes/level_two.tscn")

8.13.1 Detailed Description

Der NavigationManager ist für das Laden von Leveln und das Spawnen des Spielers verantwortlich. Der NavigationManager ist ein Singleton, der in der Haupt-Szene platziert wird und von anderen Skripten verwendet wird, um Level zu laden und den Spieler zu spawnen.

Definition at line 7 of file NavigationManager.cs.

8.13.2 Member Function Documentation

8.13.2.1 _Ready()

```
override void NavigationManager._Ready ()  [inline]
```

Initialisiert den NavigationManager und setzt ihn als Singleton.

Definition at line 32 of file NavigationManager.cs.

```
00032                                {
00033         Instance = this;
00034     }
```

References Instance.

8.13.2.2 DeferredChangeScene()

```
void NavigationManager.DeferredChangeScene (
             PackedScene SceneToLoad)  [inline], [private]
```

Diese Methode wird aufgerufen, um die Szene zu wechseln.

Parameters

SceneToLoad	Die Szene, die geladen werden soll.

Definition at line 83 of file NavigationManager.cs.

```
00084     {
00085         GetTree().ChangeSceneToPacked(SceneToLoad);
00086     }
```

Referenced by GoToLevel().

8.13.2.3 GoToLevel()

```
void NavigationManager.GoToLevel (
            string LevelTag,
            string DestinationTag)  [inline]
```

Lädt das angegebene Level und setzt das Ziel-Tag für den Spieler-Spawn.

Parameters

LevelTag	Das Tag des Levels, das geladen werden soll.
DestinationTag	Das Tag der Tür, an der der Spieler spawnen soll.

Definition at line 41 of file NavigationManager.cs.

```
00042     {
00043         PackedScene SceneToLoad = null;
00044
00045         // Bestimmen, welches Level geladen werden soll
00046         switch (LevelTag)
00047         {
00048             case "main_menu":
00049                 SceneToLoad = SceneMainMenu;
00050                 break;
00051             case "intro":
00052                 SceneToLoad = SceneIntro;
00053                 break;
00054             case "level1":
00055                 SceneToLoad = SceneLevel1;
00056                 break;
00057             case "bossRoom":
00058                 SceneToLoad = SceneBoss;
00059                 break;
00060             case "level_one":
00061                 SceneToLoad = SceneLevelOne;
00062                 break;
00063             case "level_two":
00064                 SceneToLoad = SceneLevelTwo;
00065                 break;
00066         }
00067
00068         // Überprüfen, ob eine Szene ausgewählt wurde und diese dann laden
00069         if (SceneToLoad != null){
00070             if(SceneToLoad != SceneMainMenu){
00071                 PlayerStats.Instance.SetCurrentLevelTag(LevelTag);
00072                 SpawnDoorTag = DestinationTag;
00073             }
00074             // Verwendung der ChangeSceneToPacked-Methode in Godot 4
00075             CallDeferred(nameof(DeferredChangeScene), SceneToLoad);
00076         }
00077     }
```

References DeferredChangeScene(), PlayerStats.Instance, SceneBoss, SceneIntro, SceneLevel1, SceneLevelOne, SceneLevelTwo, SceneMainMenu, PlayerStats.SetCurrentLevelTag(), and SpawnDoorTag.

Referenced by MainMenu.OnContinueButtonPressed(), Door.OnPlayerBodyEntered(), MainMenu.OnSave1SelectPressed(), MainMenu.OnSave2SelectPressed(), MainMenu.OnSave3SelectPressed(), Hud.OnSaveMenuButtonPressed(), and Player.Respawn().

8.13.2.4 OnTriggerPlayerSpawnEventHandler()

```
delegate void NavigationManager.OnTriggerPlayerSpawnEventHandler (
            Vector2 Position,
            string Direction)
```

Das Signal, das ausgelöst wird, wenn der Spieler spawnen soll.

Parameters

Position	Die Position, an der der Spieler spawnen soll.
Direction	Die Richtung, in die der Spieler schauen soll.

8.13.2.5 TriggerPlayerSpawn()

```
void NavigationManager.TriggerPlayerSpawn (
            Vector2 Position,
            string Direction)  [inline]
```

Lädt das angegebene Level und setzt das Ziel-Tag für den Spieler-Spawn.

Parameters

Position	Die Position, an der der Spieler spawnen soll.
Direction	Die Richtung, in die der Spieler schauen soll.

Definition at line 93 of file NavigationManager.cs.

```
00094     {
00095         EmitSignal(SignalName.OnTriggerPlayerSpawn, Position, Direction);
00096     }
```

8.13.3 Member Data Documentation

8.13.3.1 SceneBoss

```
readonly PackedScene NavigationManager.SceneBoss = (PackedScene)GD.Load("res://Scenes/boss↩
Room.tscn")  [static], [private]
```

Definition at line 14 of file NavigationManager.cs.

Referenced by GoToLevel().

8.13.3.2 SceneIntro

```
readonly PackedScene NavigationManager.SceneIntro = (PackedScene)GD.Load("res://Scenes/intro.↩
tscn")  [static], [private]
```

Definition at line 12 of file NavigationManager.cs.

Referenced by GoToLevel().

8.13.3.3 SceneLevel1

```
readonly PackedScene NavigationManager.SceneLevel1 = (PackedScene)GD.Load("res://Scenes/level1.↩
tscn")  [static], [private]
```

Definition at line 13 of file NavigationManager.cs.

Referenced by GoToLevel().

8.13.3.4 SceneLevelOne

```
readonly PackedScene NavigationManager.SceneLevelOne = (PackedScene)GD.Load("res://Scenes/level↩
_one.tscn")  [static], [private]
```

Definition at line 15 of file NavigationManager.cs.

Referenced by GoToLevel().

8.13.3.5 SceneLevelTwo

```
readonly PackedScene NavigationManager.SceneLevelTwo = (PackedScene)GD.Load("res://Scenes/level↩
_two.tscn")  [static], [private]
```

Definition at line 16 of file NavigationManager.cs.

Referenced by GoToLevel().

8.13.3.6 SceneMainMenu

```
readonly PackedScene NavigationManager.SceneMainMenu = (PackedScene)GD.Load("res://Scenes/main↩
_menu.tscn")  [static], [private]
```

Definition at line 11 of file NavigationManager.cs.

Referenced by GoToLevel().

8.13.4 Property Documentation

8.13.4.1 Instance

```
NavigationManager NavigationManager.Instance  [static], [get], [private set]
```

Definition at line 9 of file NavigationManager.cs.

```
00009 { get; private set; }
```

Referenced by _Ready(), MainMenu.OnContinueButtonPressed(), MainMenu.OnSave1SelectPressed(), MainMenu.OnSave2SelectPressed(), MainMenu.OnSave3SelectPressed(), and Hud.OnSaveMenuButtonPressed().

8.13.4.2 SpawnDoorTag

```
string NavigationManager.SpawnDoorTag  [get], [private set]
```

Definition at line 19 of file NavigationManager.cs.

```
00019 { get; private set; }
```

Referenced by LevelManager._Ready(), and GoToLevel().

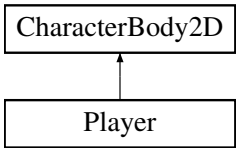
The documentation for this class was generated from the following file:

- C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/NavigationManager.cs

8.14 Player Class Reference

Klasse für den Spielercharakter. Verwaltet Bewegung, Sprünge, Angriffe und Animationen.

Inheritance diagram for Player:



Public Member Functions

- override void _Ready ()

  Initialisierung der Referenzen. Findet die relevanten Knoten in der Szene und weist sie zu.
- override void _PhysicsProcess (double DeltaTime)

  Physikalische Prozesse werden in jedem Frame ausgeführt. Berechnet Gravitation, Bewegung, Sprünge und Dashes.
- void MaxHeal ()

  Heilt den Spieler vollständig, indem die aktuellen Lebenspunkte auf das Maximum gesetzt werden.
- void TakeDamage (Damage Damage)

  Wendet Schaden auf den Spieler an. Reduziert die aktuellen Lebenspunkte basierend auf dem übergebenen Schaden und wendet einen Rückstoßeffekt an.
- Damage GetDamage ()

  Gibt den Schaden zurück, den der Spieler mit seinem aktuellen Angriff verursacht. Der Schaden basiert auf der letzten Angriffsmethode (`light_attack` oder `heavy_attack`).
- void RegenerateStamina (float Amount, double delta)

  Regeneriert die Stamina des Spielers, wenn er für eine bestimmte Zeit keine Stamina-verbrauchende Aktion durchgeführt hat.
- bool UseStamina (float Amount)

  Verbraucht eine bestimmte Menge an Stamina, falls genügend verfügbar ist. Setzt den Inaktivitäts-Timer zurück, wenn Stamina verbraucht wird.
- void SlowPlayer (float SlowAmount)

  Verlangsamt den Spieler um einen bestimmten Prozentsatz.
- void Respawn ()

  Lässt den Spieler am Checkpoint spawnen.
- BloodVial GetBloodVials ()

  Getter für BloodVials.
- void SetSinAmount (int Value)

  Setzt den SinAmount des Spielers.

Private Member Functions

- void HandleJump ()

  Verarbeitet die Sprunglogik. Setzt den Sprungzähler zurück und ermöglicht einen Doppelsprung.
- void HandleMovement (double DeltaTime)

  Verarbeitet die Bewegung des Spielers. Regelt normale Bewegungen, Dashes und Kollisionen.
- void StartDash ()

  Startet den Dash-Prozess.
- void DashInProgress (double DeltaTime)

  Führt die Logik während eines Dashes aus.
- void CreateDashEffect ()

  Erstellt einen visuellen Dash-Trail. Der Spieler hinterlässt eine Spur während des Dashes.
- void StopDash ()

  Stoppt den Dash.
- bool IsAttacking ()

  Überprüft, ob der Spieler gerade angreift.
- bool IsBlocking ()

  Überprüft, ob der Spieler blockiert.
- void OnSpawn (Vector2 position, string direction)

  Wird aufgerufen, wenn der Spieler an einer neuen Position spawnen soll.
- void UpdateAnimations ()

  Aktualisiert die Animationen des Spielers.

Private Attributes

- int JumpMax = 2
- int JumpCount = 0
- Vector2 DashDirection = Vector2.Zero
- float DashSpeed = 300f
- bool IsDashing = false
- bool CanDash = true
- float DashTrailInterval = 0.05f
- float DashTrailTimer = 0f
- AnimationPlayer AnimationPlayer
- Sprite2D Sprite
- Timer DashEffect
- Timer DashTimer
- CollisionShape2D SwordCollision
- CollisionShape2D PlayerHitbox
- BloodVial BloodVials
- Label SinDisplay
- Vector2 HauptHitbox
- int LastAttack = 0
- float TimeSinceLastStaminaUse = 0f

Static Private Attributes

- const float SPEED = 100f
- const float JUMP_VELOCITY = -300f

8.14.1 Detailed Description

Klasse für den Spielercharakter. Verwaltet Bewegung, Sprünge, Angriffe und Animationen.

Definition at line 8 of file Player.cs.

8.14.2 Member Function Documentation

8.14.2.1 _PhysicsProcess()

```
override void Player._PhysicsProcess (
            double DeltaTime)  [inline]
```

Physikalische Prozesse werden in jedem Frame ausgeführt. Berechnet Gravitation, Bewegung, Sprünge und Dashes.

Parameters

DeltaTime	Zeit seit dem letzten Frame.

Definition at line 67 of file Player.cs.

```
00067                                                       {
00068         // Gravitation hinzufügen, wenn der Charakter nicht am Boden ist
00069         if (!IsOnFloor()) {
00070             Velocity += GetGravity() * (float)DeltaTime;
00071         } else {
00072             CanDash = true; // Dash wird zurückgesetzt, wenn der Charakter am Boden ist
00073         }
00074 
00075         TimeSinceLastStaminaUse += (float)DeltaTime;
00076         RegenerateStamina(20f, DeltaTime);
00077 
00078         // Heal
00079         if(Input.IsActionJustPressed("heal")){
00080             BloodVials.UseBloodVial();
00081         }
00082 
00083         HandleJump();
00084         HandleMovement(DeltaTime);
00085         MoveAndSlide();
00086         UpdateAnimations();
00087         PlayerStats.Instance.SetPosition(Position);
00088         }
```

References BloodVials, CanDash, HandleJump(), HandleMovement(), PlayerStats.Instance, RegenerateStamina(), PlayerStats.SetPosition(), TimeSinceLastStaminaUse, UpdateAnimations(), and BloodVial.UseBloodVial().

8.14.2.2 _Ready()

```
override void Player._Ready ()  [inline]
```

Initialisierung der Referenzen. Findet die relevanten Knoten in der Szene und weist sie zu.

Definition at line 43 of file Player.cs.

```
00043                                {
00044         AnimationPlayer = GetNode<AnimationPlayer>("AnimationPlayer");
00045         Sprite = GetNode<Sprite2D>("Sprite2D");
00046         DashEffect = GetNode<Timer>("DashEffect");
00047         DashTimer = GetNode<Timer>("DashTimer");
00048         SwordCollision = GetNode<CollisionShape2D>("Sprite2D/SwordHit/SwordCollision");
00049         PlayerHitbox = GetNode<CollisionShape2D>("PlayerHitbox");
00050         HauptHitbox = PlayerHitbox.Position;
00051         BloodVials = GetNode<BloodVial>("../HUD/BloodVial/Counter");
00052         SinDisplay = GetNode<Label>("../HUD/SinAmount/Counter");
00053 
00054         SinDisplay.Text = PlayerStats.Instance.GetSinAmount() + "";
00055 
00056         NavigationManager navigationManager = GetNode<NavigationManager>("/root/NavigationManager");
00057         navigationManager.Connect("OnTriggerPlayerSpawn", new Callable(this, nameof(OnSpawn)));
00058 
00059         Position = PlayerStats.Instance.GetPosition();
00060     }
```

References AnimationPlayer, BloodVials, DashEffect, DashTimer, PlayerStats.GetPosition(), PlayerStats.GetSinAmount(), HauptHitbox, PlayerStats.Instance, OnSpawn(), PlayerHitbox, SinDisplay, Sprite, and SwordCollision.

8.14.2.3 CreateDashEffect()

```
void Player.CreateDashEffect ()  [inline], [private]
```

Erstellt einen visuellen Dash-Trail. Der Spieler hinterlässt eine Spur während des Dashes.

Definition at line 207 of file Player.cs.

```
00207                                   {
00208         Sprite2D PlayerCopyNode = (Sprite2D)Sprite.Duplicate();
00209         GetParent().AddChild(PlayerCopyNode);
00210
00211         CollisionShape2D SwordCollisionCopy =
      PlayerCopyNode.GetNode<CollisionShape2D>("SwordHit/SwordCollision");
00212         if (SwordCollisionCopy != null) {
00213             SwordCollisionCopy.Disabled = true; // Deaktiviere die Kollision der Kopie
00214         }
00215
00216         PlayerCopyNode.GlobalPosition = GlobalPosition + new Vector2(0, Sprite.Texture.GetHeight() *
      Sprite.Scale.Y * -0.5f);
00217
00218         // Verblassen-Effekt für den Dash-Trail hinzufügen
00219         float AnimationTime = (float)(DashTimer.WaitTime / 3);
00220
00221         Timer FadeTimer1 = new Timer();
00222         AddChild(FadeTimer1);
00223         FadeTimer1.Timeout += () => {
00224             if (IsInstanceValid(PlayerCopyNode)) {
00225                 PlayerCopyNode.Modulate = new Color(PlayerCopyNode.Modulate, 0.4f);
00226             }
00227         };
00228         FadeTimer1.Start(AnimationTime);
00229
00230         Timer FadeTimer2 = new Timer();
00231         AddChild(FadeTimer2);
00232         FadeTimer2.Timeout += () => {
00233             if (IsInstanceValid(PlayerCopyNode)) {
00234                 PlayerCopyNode.Modulate = new Color(PlayerCopyNode.Modulate, 0.2f);
00235             }
00236         };
00237         FadeTimer2.Start(AnimationTime * 2);
00238
00239         Timer FadeTimer3 = new Timer();
00240         AddChild(FadeTimer3);
00241         FadeTimer3.Timeout += () => {
00242             if (IsInstanceValid(PlayerCopyNode)) {
00243                 PlayerCopyNode.QueueFree();
00244             }
00245         };
00246         FadeTimer3.Start(AnimationTime * 3);
00247     }
```

References DashTimer, and Sprite.

Referenced by DashInProgress().

8.14.2.4 DashInProgress()

```
void Player.DashInProgress (
             double DeltaTime)  [inline], [private]
```

Führt die Logik während eines Dashes aus.

Parameters

DeltaTime	Zeit seit dem letzten Frame.

Definition at line 187 of file Player.cs.

```
00187                                                {
00188         // Charakter bewegt sich in die Dash-Richtung mit Dash-Geschwindigkeit
00189         if (DashDirection == Vector2.Up) {
00190             Velocity = DashDirection / 1.5f * DashSpeed;
00191         } else {
00192             Velocity = DashDirection * DashSpeed;
00193         }
00194 
00195         // Dash-Trail bei Intervallen erstellen
00196         DashTrailTimer -= (float)DeltaTime;
00197         if (DashTrailTimer <= 0f) {
00198             CreateDashEffect();
00199             DashTrailTimer = DashTrailInterval;
00200         }
00201     }
```

References CreateDashEffect(), DashDirection, DashSpeed, DashTrailInterval, and DashTrailTimer.

Referenced by HandleMovement().

8.14.2.5 GetBloodVials()

```
BloodVial Player.GetBloodVials ()  [inline]
```

Getter für BloodVials.

Returns

BloodVial

Definition at line 383 of file Player.cs.

```
00383                                    {
00384         return BloodVials;
00385     }
```

References BloodVials.

Referenced by Checkpoint.OnPlayerBodyEntered().

8.14.2.6 GetDamage()

```
Damage Player.GetDamage ()  [inline]
```

Gibt den Schaden zurück, den der Spieler mit seinem aktuellen Angriff verursacht. Der Schaden basiert auf der letzten Angriffsmethode (`light_attack` oder `heavy_attack`).

Returns

Eine Instanz der Klasse `Damage`, die den physischen Schaden, wahren Schaden und Rückstoß enthält.

Definition at line 317 of file Player.cs.

```
00317                                {
00318         if(LastAttack == 1){
00319             return new Damage(50, 0, Vector2.Zero, this);
00320         }
00321         if(LastAttack == 2){
00322             Vector2 Push = new Vector2(20,0);
00323             if(Sprite.FlipH){
00324                 Push = -Push;
00325             }
00326             return new Damage(100, 0, Push, this);
00327         }
00328         return new Damage(0,0,Vector2.Zero, this);
00329     }
```

References LastAttack, and Sprite.

Referenced by BaseEnemy.OnHitboxAreaEntered().

8.14.2.7 HandleJump()

```
void Player.HandleJump ()  [inline], [private]
```

Verarbeitet die Sprunglogik. Setzt den Sprungzähler zurück und ermöglicht einen Doppelsprung.

Definition at line 94 of file Player.cs.

```
00094                               {
00095         // Sprungzähler zurücksetzen, wenn der Charakter am Boden ist
00096         if (JumpCount != 0 && IsOnFloor()) {
00097             JumpCount = 0;
00098         }
00099
00100         // Überprüfen, ob der Sprung-Button gedrückt wurde und der Charakter noch Sprünge übrig hat
00101         if (Input.IsActionJustPressed("ui_up") && JumpCount < JumpMax) {
00102             if (JumpCount == 0) {
00103             // Erster Sprung ohne Stamina-Verlust
00104             Velocity = new Vector2(Velocity.X, JUMP_VELOCITY);
00105             JumpCount += 1;
00106             } else if (JumpCount > 0) {
00107                 // Beim Doppelsprung Stamina prüfen und abziehen
00108                 if (UseStamina(15)) {
00109                     Velocity = new Vector2(Velocity.X, JUMP_VELOCITY);
00110                     JumpCount += 1;
00111                 }
00112             }
00113         }
00114     }
```

References JUMP_VELOCITY, JumpCount, JumpMax, and UseStamina().

Referenced by _PhysicsProcess().

8.14.2.8 HandleMovement()

```
void Player.HandleMovement (
             double DeltaTime)  [inline], [private]
```

Verarbeitet die Bewegung des Spielers. Regelt normale Bewegungen, Dashes und Kollisionen.

Parameters

DeltaTime	Zeit seit dem letzten Frame.

Definition at line 121 of file Player.cs.

```
00121                                                 {
00122         Vector2 direction = new Vector2(Input.GetAxis("ui_left", "ui_right"), Input.GetAxis("ui_up",
      "ui_down")).Normalized();
00123         float currentSpeed = SPEED;
00124
00125         // Sprite umdrehen basierend auf der Bewegungsrichtung und Kollision umdrehen
00126         if (direction.X < 0) {
00127             Sprite.FlipH = true;
00128             SwordCollision.Position = new Vector2(-Mathf.Abs(SwordCollision.Position.X),
      SwordCollision.Position.Y);
00129             PlayerHitbox.Position = new Vector2(Sprite.Position.X * 1.8f, PlayerHitbox.Position.Y);
00130         } else if (direction.X > 0) {
00131             Sprite.FlipH = false;
00132             SwordCollision.Position = new Vector2(Mathf.Abs(SwordCollision.Position.X),
      SwordCollision.Position.Y);
00133             PlayerHitbox.Position = HauptHitbox;
00134         }
00135
00136         // Geschwindigkeit reduzieren, wenn der Spieler angreift
00137         if (AnimationPlayer.CurrentAnimation == "light_attack") {
00138             currentSpeed *= 0.5f;
00139         } else if (AnimationPlayer.CurrentAnimation == "heavy_attack") {
00140             currentSpeed *= 0.15f;
00141         }
00142
00143         // Blockieren stoppt die Bewegung
```

```
00144         if (IsBlocking()) {
00145             currentSpeed = 0;
00146         }
00147 
00148         if (IsDashing) {
00149             DashInProgress(DeltaTime);
00150         } else {
00151             // Normale Bewegung verarbeiten, wenn kein Dash aktiv ist
00152             if (direction != Vector2.Zero) {
00153                 Velocity = new Vector2(direction.X * currentSpeed, Velocity.Y);
00154             } else {
00155                 Velocity = new Vector2(Mathf.MoveToward(Velocity.X, 0, SPEED), Velocity.Y);
00156             }
00157 
00158             // Überprüfen, ob der Dash-Button gedrückt wurde mit eine Bewegungsrichtung und nicht
      schon am angreifen ist
00159             if (Input.IsActionJustPressed("dash") && direction != Vector2.Zero && CanDash &&
      !IsAttacking()) {
00160                 // Wenn der Player genug Stamina hat kann er dashen
00161                 if (UseStamina(20)){
00162                     DashDirection = direction;
00163                     StartDash();
00164                 }
00165             }
00166         }
00167     }
```

References AnimationPlayer, CanDash, DashDirection, DashInProgress(), HauptHitbox, IsAttacking(), IsBlocking(), IsDashing, PlayerHitbox, SPEED, Sprite, StartDash(), SwordCollision, and UseStamina().

Referenced by _PhysicsProcess().

8.14.2.9 IsAttacking()

```
bool Player.IsAttacking ()  [inline], [private]
```

Überprüft, ob der Spieler gerade angreift.

Returns

true, wenn der Spieler angreift.

Definition at line 265 of file Player.cs.

```
00265                                {
00266         return AnimationPlayer.CurrentAnimation == "heavy_attack" || AnimationPlayer.CurrentAnimation
      == "light_attack";
00267     }
```

Referenced by HandleMovement(), and UpdateAnimations().

8.14.2.10 IsBlocking()

```
bool Player.IsBlocking ()  [inline], [private]
```

Überprüft, ob der Spieler blockiert.

Returns

true, wenn der Spieler blockiert.

Definition at line 273 of file Player.cs.

```
00273                               {
00274         return AnimationPlayer.CurrentAnimation == "block";
00275     }
```

Referenced by HandleMovement(), TakeDamage(), and UpdateAnimations().

8.14.2.11 MaxHeal()

```
void Player.MaxHeal ()  [inline]
```

Heilt den Spieler vollständig, indem die aktuellen Lebenspunkte auf das Maximum gesetzt werden.

Definition at line 280 of file Player.cs.

```
00280                       {
00281         PlayerStats.Instance.SetCurrentHealth(PlayerStats.Instance.GetMaxHealthPoints());
00282     }
```

References PlayerStats.GetMaxHealthPoints(), PlayerStats.Instance, and PlayerStats.SetCurrentHealth().

Referenced by Checkpoint.OnPlayerBodyEntered().

8.14.2.12 OnSpawn()

```
void Player.OnSpawn (
            Vector2 position,
            string direction)  [inline], [private]
```

Wird aufgerufen, wenn der Spieler an einer neuen Position spawnen soll.

Parameters

position	Die Position, an der der Spieler spawnen soll.
direction	Die Richtung, in die der Spieler schauen soll.

Definition at line 402 of file Player.cs.

```
00402                                                       {
00403
00404         // Spielerposition auf die übergebene Position setzen
00405         if (direction == "right")
00406         {
00407             // Update the x value by adding 50 to it, keep the original y value
00408             Sprite.FlipH = false;
00409             position = position with { X = position.X + 25 };
00410         }
00411         else if (direction == "left")
00412         {
00413             // Update the x value by subtracting 50 from it, keep the original y value
00414             Sprite.FlipH = true;
00415             position = position with { X = position.X - 25 };
00416         }
00417         Position = position;
00418
00419     }
```

Referenced by _Ready().

8.14.2.13 RegenerateStamina()

```
void Player.RegenerateStamina (
            float Amount,
            double delta)  [inline]
```

Regeneriert die Stamina des Spielers, wenn er für eine bestimmte Zeit keine Stamina-verbrauchende Aktion durchgeführt hat.

Parameters

Amount	Menge der Stamina, die regeneriert werden soll.
delta	Zeit seit dem letzten Frame.

Definition at line 336 of file Player.cs.

```
00336                                                       {
00337         // Wenn die Verzögerungszeit erreicht wurde, regeneriere Stamina
00338         if (TimeSinceLastStaminaUse >= 1f) {
00339             PlayerStats.Instance.SetStamina(PlayerStats.Instance.GetStamina() + Amount *
      (float)delta); // Regeneriere Stamina abhängig von der Zeit
00340         }
00341     }
```

References PlayerStats.GetStamina(), PlayerStats.Instance, PlayerStats.SetStamina(), and TimeSinceLastStaminaUse.

Referenced by _PhysicsProcess().

8.14.2.14 Respawn()

```
void Player.Respawn ()  [inline]
```

Lässt den Spieler am Checkpoint spawnen.

Definition at line 372 of file Player.cs.

```
00372                       {
00373         var NavigationManager = GetNode<NavigationManager>("/root/NavigationManager");
00374         NavigationManager.GoToLevel(PlayerStats.Instance.GetRespawnLevelTag(), "spawn");
00375         BloodVials.ResetUses();
00376
00377     }
```

References BloodVials, PlayerStats.GetRespawnLevelTag(), NavigationManager.GoToLevel(), PlayerStats.Instance, and BloodVial.ResetUses().

Referenced by TakeDamage().

8.14.2.15 SetSinAmount()

```
void Player.SetSinAmount (
            int Value)  [inline]
```

Setzt den SinAmount des Spielers.

Parameters

Value	Der neue Wert für den SinAmount.

Definition at line 391 of file Player.cs.

```
00391                                       {
00392         // SinAmount muss immer >= 0 sein
00393         PlayerStats.Instance.SetSinAmount(Value);
00394         SinDisplay.Text = PlayerStats.Instance.GetSinAmount() + "";
00395     }
```

References PlayerStats.GetSinAmount(), PlayerStats.Instance, and PlayerStats.SetSinAmount().

Referenced by BaseEnemy.Die().

8.14.2.16 SlowPlayer()

```
void Player.SlowPlayer (
            float SlowAmount)  [inline]
```

Verlangsamt den Spieler um einen bestimmten Prozentsatz.

Parameters

SlowAmount	Der Prozentsatz, um den der Spieler verlangsamt werden soll.

Definition at line 365 of file Player.cs.

```
00365                                           {
00366         Velocity = new Vector2(Velocity.X * SlowAmount, Velocity.Y);
00367     }
```

Referenced by Spike.OnPlayerBodyEntered(), and SpikeDynamic.OnPlayerBodyEntered().

8.14.2.17 StartDash()

```
void Player.StartDash ()  [inline], [private]
```

Startet den Dash-Prozess.

Definition at line 172 of file Player.cs.

```
00172                               {
00173         SetCollisionLayerValue(1,false);
00174         SetCollisionMaskValue(1,false);
00175         IsDashing = true;
00176         CanDash = false;
00177         DashTimer.Timeout += StopDash;
00178         DashTimer.Start();
00179         DashEffect.Start();
00180         DashTrailTimer = 0f;
00181     }
```

References CanDash, DashEffect, DashTimer, DashTrailTimer, IsDashing, and StopDash().

Referenced by HandleMovement().

8.14.2.18 StopDash()

```
void Player.StopDash ()  [inline], [private]
```

Stoppt den Dash.

Definition at line 252 of file Player.cs.

```
00252                               {
00253         IsDashing = false;
00254         DashEffect.Stop();
00255         DashTimer.Stop();
00256         DashTimer.Timeout -= StopDash;
00257         SetCollisionLayerValue(1,true);
00258         SetCollisionMaskValue(1,true);
00259     }
```

References DashEffect, DashTimer, IsDashing, and StopDash().

Referenced by StartDash(), and StopDash().

8.14.2.19 TakeDamage()

```
void Player.TakeDamage (
            Damage Damage)  [inline]
```

Wendet Schaden auf den Spieler an. Reduziert die aktuellen Lebenspunkte basierend auf dem übergebenen Schaden und wendet einen Rückstoßeffekt an.

Parameters

Damage	Instanz der Klasse `Damage`, die den physischen und wahren Schaden sowie den Rückstoß enthält.

Definition at line 289 of file Player.cs.

```
00289                                       {
00290         float totalDamage = Damage.GetTrueDMG();
00291         if(!IsBlocking()){
00292             totalDamage += Damage.GetPhysicalDMG();
00293         } else {
00294             float CurrentStamina = PlayerStats.Instance.GetStamina();
00295             CurrentStamina -= Damage.GetPhysicalDMG();
00296             if(CurrentStamina < 0){
00297                 totalDamage -= CurrentStamina;
00298             }
00299             PlayerStats.Instance.SetStamina(CurrentStamina);
00300         }
00301
00302         PlayerStats.Instance.SetCurrentHealth(PlayerStats.Instance.GetCurrentHealth() - totalDamage);
00303         Position += Damage.GetPushAmount();
00304
00305         // Überprüfe, ob der Spieler gestorben ist
00306         if (PlayerStats.Instance.GetCurrentHealth() <= 0){
00307             GD.Print("Spieler ist gestorben!");
00308             Respawn();
00309         }
00310     }
```

References PlayerStats.GetCurrentHealth(), Damage.GetPhysicalDMG(), Damage.GetPushAmount(), PlayerStats.GetStamina(), Damage.GetTrueDMG(), PlayerStats.Instance, IsBlocking(), Respawn(), PlayerStats.SetCurrentHealth(), and PlayerStats.SetStamina().

Referenced by BaseEnemy.CheckPlayerHit(), Spike.OnPlayerBodyEntered(), SpikeDynamic.OnPlayerBodyEntered(), Spike.OnTimerTimeout(), and SpikeDynamic.OnTimerTimeout().

8.14.2.20 UpdateAnimations()

```
void Player.UpdateAnimations ()  [inline], [private]
```

Aktualisiert die Animationen des Spielers.

Definition at line 425 of file Player.cs.

```
00425                                     {
00426         if (Input.IsActionJustPressed("light_attack") && !IsDashing && !IsAttacking()) {
00427             if (UseStamina(10)){
00428                 LastAttack = 1;
00429                 AnimationPlayer.Play("light_attack");
00430             }
00431         } else if (Input.IsActionJustPressed("heavy_attack") && !IsDashing && !IsAttacking()) {
00432             if (UseStamina(25)){
00433                 LastAttack = 2;
00434                 AnimationPlayer.Play("heavy_attack");
00435             }
00436         }
00437         if (Input.IsActionPressed("block") && !IsDashing && !IsAttacking() && IsOnFloor()) {
00438             if (UseStamina(0)){
00439                 AnimationPlayer.Play("block");
00440                 LastAttack = 0;
00441             }
00442         }
00443
00444         if (IsOnFloor() && !IsAttacking() && !IsBlocking()) {
00445             LastAttack = 0;
00446             if (Velocity.X == 0) {
00447                 AnimationPlayer.Play("idle");
00448             } else {
00449                 AnimationPlayer.Play("run");
00450             }
00451         } else if (!IsOnFloor() && !IsAttacking() && !IsBlocking()) {
00452             LastAttack = 0;
00453             if (Velocity.Y < 0) {
00454                 AnimationPlayer.Play("jump");
00455             } else if (Velocity.Y > 0) {
```

```
00456                 AnimationPlayer.Play("fall");
00457             }
00458         }
00459     }
```

References AnimationPlayer, IsAttacking(), IsBlocking(), IsDashing, LastAttack, and UseStamina().

Referenced by _PhysicsProcess().

8.14.2.21 UseStamina()

```
bool Player.UseStamina (
            float Amount)  [inline]
```

Verbraucht eine bestimmte Menge an Stamina, falls genügend verfügbar ist. Setzt den Inaktivitäts-Timer zurück, wenn Stamina verbraucht wird.

Parameters

Amount	Die Menge an Stamina, die verbraucht werden soll.

Returns

`true`, wenn genügend Stamina verfügbar war und die Aktion ausgeführt wurde; andernfalls `false`.

Definition at line 349 of file Player.cs.

```
00349                                       {
00350         // Versucht, eine bestimmte Menge an Stamina zu verbrauchen.
00351         // Gibt true zurück, wenn genug Stamina verfügbar war; andernfalls false.
00352         if (PlayerStats.Instance.GetStamina() >= Amount) {
00353             PlayerStats.Instance.SetStamina(PlayerStats.Instance.GetStamina() - Amount);
00354             TimeSinceLastStaminaUse = 0f;
00355             return true;
00356         }
00357 
00358         return false;
00359     }
```

References PlayerStats.GetStamina(), PlayerStats.Instance, PlayerStats.SetStamina(), and TimeSinceLastStaminaUse.

Referenced by HandleJump(), HandleMovement(), and UpdateAnimations().

8.14.3 Member Data Documentation

8.14.3.1 AnimationPlayer

```
AnimationPlayer Player.AnimationPlayer  [private]
```

Definition at line 24 of file Player.cs.

Referenced by _Ready(), HandleMovement(), and UpdateAnimations().

8.14.3.2 BloodVials

```
BloodVial Player.BloodVials  [private]
```

Definition at line 30 of file Player.cs.

Referenced by _PhysicsProcess(), _Ready(), GetBloodVials(), and Respawn().

8.14.3.3 CanDash

```
bool Player.CanDash = true  [private]
```

Definition at line 19 of file Player.cs.

Referenced by _PhysicsProcess(), HandleMovement(), and StartDash().

8.14.3.4 DashDirection

```
Vector2 Player.DashDirection = Vector2.Zero  [private]
```

Definition at line 16 of file Player.cs.

Referenced by DashInProgress(), and HandleMovement().

8.14.3.5 DashEffect

```
Timer Player.DashEffect  [private]
```

Definition at line 26 of file Player.cs.

Referenced by _Ready(), StartDash(), and StopDash().

8.14.3.6 DashSpeed

```
float Player.DashSpeed = 300f  [private]
```

Definition at line 17 of file Player.cs.

Referenced by DashInProgress().

8.14.3.7 DashTimer

```
Timer Player.DashTimer  [private]
```

Definition at line 27 of file Player.cs.

Referenced by _Ready(), CreateDashEffect(), StartDash(), and StopDash().

8.14.3.8 DashTrailInterval

```
float Player.DashTrailInterval = 0.05f  [private]
```

Definition at line 20 of file Player.cs.

Referenced by DashInProgress().

8.14.3.9 DashTrailTimer

```
float Player.DashTrailTimer = 0f  [private]
```

Definition at line 21 of file Player.cs.

Referenced by DashInProgress(), and StartDash().

8.14.3.10 HauptHitbox

```
Vector2 Player.HauptHitbox  [private]
```

Definition at line 33 of file Player.cs.

Referenced by _Ready(), and HandleMovement().

8.14.3.11 IsDashing

```
bool Player.IsDashing = false  [private]
```

Definition at line 18 of file Player.cs.

Referenced by HandleMovement(), StartDash(), StopDash(), and UpdateAnimations().

8.14.3.12 JUMP_VELOCITY

```
const float Player.JUMP_VELOCITY = -300f  [static], [private]
```

Definition at line 12 of file Player.cs.

Referenced by HandleJump().

8.14.3.13 JumpCount

```
int Player.JumpCount = 0  [private]
```

Definition at line 14 of file Player.cs.

Referenced by HandleJump().

8.14.3.14 JumpMax

```
int Player.JumpMax = 2  [private]
```

Definition at line 13 of file Player.cs.

Referenced by HandleJump().

8.14.3.15 LastAttack

```
int Player.LastAttack = 0  [private]
```

Definition at line 34 of file Player.cs.

Referenced by GetDamage(), and UpdateAnimations().

8.14.3.16 PlayerHitbox

```
CollisionShape2D Player.PlayerHitbox  [private]
```

Definition at line 29 of file Player.cs.

Referenced by _Ready(), and HandleMovement().

8.14.3.17 SinDisplay

```
Label Player.SinDisplay  [private]
```

Definition at line 31 of file Player.cs.

Referenced by _Ready().

8.14.3.18 SPEED

```
const float Player.SPEED = 100f  [static], [private]
```

Definition at line 11 of file Player.cs.

Referenced by HandleMovement().

8.14.3.19 Sprite

```
Sprite2D Player.Sprite  [private]
```

Definition at line 25 of file Player.cs.

Referenced by _Ready(), CreateDashEffect(), GetDamage(), and HandleMovement().

8.14.3.20 SwordCollision

```
CollisionShape2D Player.SwordCollision  [private]
```

Definition at line 28 of file Player.cs.

Referenced by _Ready(), and HandleMovement().

8.14.3.21 TimeSinceLastStaminaUse

```
float Player.TimeSinceLastStaminaUse = 0f  [private]
```

Definition at line 37 of file Player.cs.

Referenced by _PhysicsProcess(), RegenerateStamina(), and UseStamina().

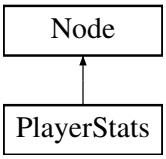
The documentation for this class was generated from the following file:

- C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Player.cs

8.15 PlayerStats Class Reference

Klasse für die Spielerstats.

Inheritance diagram for PlayerStats:



Public Member Functions

- override void _Ready ()
  Initialisierung der Referenzen.
- String GetRespawnLevelTag ()
  Getter für RespawnLevelTag.
- void SetRespawnLevelTag (String levelTag)
  Setter für RespawnLevelTag.
- String GetCurrentLevelTag ()
  Getter für CurrentLevelTag.
- void SetCurrentLevelTag (String levelTag)
  Setter für CurrentLevelTag.
- void SetSpawnPoint (Vector2 spawnPoint)
  Setzt den SpawnPoint des Spielers.
- Vector2 GetSpawnPoint ()
  Getter für den SpawnPoint.
- void SetPosition (Vector2 position)
  Setzt die Position des Spielers.
- Vector2 GetPosition ()
  Getter für die Position.
- int GetSinAmount ()
  Getter für SinAmount.
- void SetSinAmount (int Value)
  Setzt den SinAmount des Spielers.
- float GetMaxHealthPoints ()

Gibt die maximalen Lebenspunkte des Spielers zurück.

- void SetMaxHealthPoints (float maxHealthPoints)

  Setzt die maximalen Lebenspunkte des Spielers.
- float GetCurrentHealth ()

  Gibt die aktuellen Lebenspunkte des Spielers zurück.
- void SetCurrentHealth (float Health)

  Setzt die aktuellen Lebenspunkte des Spielers.
- void SetMaxStamina (float Value)

  Setzt die maximale Stamina des Spielers.
- float GetMaxStamina ()

  Gibt die maximale Stamina des Spielers zurück.
- void SetStamina (float Value)

  Setzt die Stamina des Spielers.
- float GetStamina ()

  Gibt die aktuelle Stamina des Spielers zurück.
- void SetBVHealAmount (int Value)

  Setzt den HealAmount eines Bloodvials.
- int GetBVHealAmount ()

  Gibt den aktuellen HealAmount eines Bloodvials zurück.
- void SetBVMaxUses (int Value)

  Setzt die MaxUses der Bloodvials.
- int GetBVMaxUses ()

  Gibt die MaxUses der Bloodvials zurück.
- void SetBVCurrentUses (int Value)

  Setzt die CurrentUses der Bloodvials.
- int GetBVCurrentUses ()

  Gibt die CurrentUses der Bloodvials zurück.
- void Reload ()

  Setzt die Attribute zurück.

Properties

- static PlayerStats Instance `[get, private set]`

Private Attributes

- String RespawnLevelTag = "intro"
- String CurrentLevelTag = "intro"
- Vector2 SpawnPoint
- Vector2 Position = new Vector2(-540, 160)
- int SinAmount
- float MaxHealthPoints = 100f
- float CurrentHealth
- float MaxStamina = 100f
- float CurrentStamina
- int BVHealAmount = 25
- int BVMaxUses = 5
- int BVCurrentUses

8.15.1 Detailed Description

Klasse für die Spielerstats.

Definition at line 7 of file PlayerStats.cs.

8.15.2 Member Function Documentation

8.15.2.1 _Ready()

```
override void PlayerStats._Ready ()  [inline]
```

Initialisierung der Referenzen.

Definition at line 29 of file PlayerStats.cs.

```
00029                                {
00030         CurrentHealth = MaxHealthPoints;
00031         CurrentStamina = MaxStamina;
00032         BVCurrentUses = BVMaxUses;
00033         Instance = this;
00034     }
```

References BVCurrentUses, BVMaxUses, CurrentHealth, CurrentStamina, Instance, MaxHealthPoints, and MaxStamina.

Referenced by Reload().

8.15.2.2 GetBVCurrentUses()

```
int PlayerStats.GetBVCurrentUses ()  [inline]
```

Gibt die CurrentUses der Bloodvials zurück.

Returns

Die aktuellen CurrentUses.

Definition at line 230 of file PlayerStats.cs.

```
00230                                    {
00231         return BVCurrentUses;
00232     }
```

References BVCurrentUses.

Referenced by BloodVial._Ready(), BloodVial.ResetUses(), StorageManager.SaveGameFile(), and BloodVial.UseBloodVial().

8.15.2.3 GetBVHealAmount()

```
int PlayerStats.GetBVHealAmount ()  [inline]
```

Gibt den aktuellen HealAmount eines Bloodvials zurück.

Returns

Der aktuelle HealAmount.

Definition at line 198 of file PlayerStats.cs.

```
00198                                {
00199         return BVHealAmount;
00200     }
```

References BVHealAmount.

Referenced by BloodVial.LevelHealAmount(), StorageManager.SaveGameFile(), and BloodVial.UseBloodVial().

8.15.2.4 GetBVMaxUses()

```
int PlayerStats.GetBVMaxUses ()  [inline]
```

Gibt die MaxUses der Bloodvials zurück.

Returns

Die aktuellen MaxUses.

Definition at line 214 of file PlayerStats.cs.

```
00214                                {
00215         return BVMaxUses;
00216     }
```

References BVMaxUses.

Referenced by BloodVial.AddMaxUses(), BloodVial.ResetUses(), and StorageManager.SaveGameFile().

8.15.2.5 GetCurrentHealth()

```
float PlayerStats.GetCurrentHealth ()  [inline]
```

Gibt die aktuellen Lebenspunkte des Spielers zurück.

Returns

Die aktuellen Lebenspunkte.

Definition at line 139 of file PlayerStats.cs.

```
00139                                    {
00140         return CurrentHealth;
00141     }
```

References CurrentHealth.

Referenced by HealthBar._Process(), HealthBar._Ready(), StorageManager.SaveGameFile(), Player.TakeDamage(), and BloodVial.UseBloodVial().

8.15.2.6 GetCurrentLevelTag()

```
String PlayerStats.GetCurrentLevelTag ()  [inline]
```

Getter für CurrentLevelTag.

Returns

String CurrentLevelTag

Definition at line 56 of file PlayerStats.cs.

```
00056                                    {
00057         return CurrentLevelTag;
00058     }
```

References CurrentLevelTag.

Referenced by MainMenu.OnContinueButtonPressed(), MainMenu.OnSave1SelectPressed(), MainMenu.OnSave2SelectPressed(), MainMenu.OnSave3SelectPressed(), and StorageManager.SaveGameFile().

8.15.2.7 GetMaxHealthPoints()

```
float PlayerStats.GetMaxHealthPoints ()  [inline]
```

Gibt die maximalen Lebenspunkte des Spielers zurück.

Returns

Die maximalen Lebenspunkte.

Definition at line 122 of file PlayerStats.cs.

```
00122                                       {
00123         return MaxHealthPoints;
00124     }
```

References MaxHealthPoints.

Referenced by HealthBar._Ready(), Player.MaxHeal(), and StorageManager.SaveGameFile().

8.15.2.8 GetMaxStamina()

```
float PlayerStats.GetMaxStamina ()  [inline]
```

Gibt die maximale Stamina des Spielers zurück.

Returns

Die maximale Stamina.

Definition at line 165 of file PlayerStats.cs.

```
00165                                     {
00166         return MaxStamina;
00167     }
```

References MaxStamina.

Referenced by StaminaBar._Ready(), Checkpoint.OnPlayerBodyEntered(), and StorageManager.SaveGameFile().

8.15.2.9 GetPosition()

```
Vector2 PlayerStats.GetPosition ()  [inline]
```

Getter für die Position.

Returns

Position des Spielers

Definition at line 96 of file PlayerStats.cs.

```
00096                                   {
00097         return Position;
00098     }
```

References Position.

Referenced by LevelManager._Ready(), Player._Ready(), and StorageManager.SaveGameFile().

8.15.2.10 GetRespawnLevelTag()

```
String PlayerStats.GetRespawnLevelTag ()  [inline]
```

Getter für RespawnLevelTag.

Returns

String RespawnLevelTag

Definition at line 40 of file PlayerStats.cs.

```
00040                                       {
00041         return RespawnLevelTag;
00042     }
```

References RespawnLevelTag.

Referenced by Checkpoint.OnPlayerBodyEntered(), Player.Respawn(), and StorageManager.SaveGameFile().

8.15.2.11 GetSinAmount()

```
int PlayerStats.GetSinAmount ()  [inline]
```

Getter für SinAmount.

Returns

int Sins

Definition at line 105 of file PlayerStats.cs.

```
00105                                   {
00106         return SinAmount;
00107     }
```

References SinAmount.

Referenced by Player._Ready(), BaseEnemy.Die(), StorageManager.SaveGameFile(), and Player.SetSinAmount().

8.15.2.12 GetSpawnPoint()

```
Vector2 PlayerStats.GetSpawnPoint ()  [inline]
```

Getter für den SpawnPoint.

Returns

Der SpawnPoint des Spielers

Definition at line 80 of file PlayerStats.cs.

```
00080                                     {
00081         return SpawnPoint;
00082     }
```

References SpawnPoint.

Referenced by StorageManager.SaveGameFile().

8.15.2.13 GetStamina()

```
float PlayerStats.GetStamina ()  [inline]
```

Gibt die aktuelle Stamina des Spielers zurück.

Returns

Die aktuelle Stamina.

Definition at line 182 of file PlayerStats.cs.

```
00182                                {
00183         return CurrentStamina;
00184     }
```

References CurrentStamina.

Referenced by StaminaBar._Process(), StaminaBar._Ready(), Player.RegenerateStamina(), StorageManager.SaveGameFile(), Player.TakeDamage(), and Player.UseStamina().

8.15.2.14 Reload()

```
void PlayerStats.Reload ()  [inline]
```

Setzt die Attribute zurück.

Definition at line 237 of file PlayerStats.cs.

```
00237                        {
00238         Instance = new PlayerStats();
00239         Instance._Ready();
00240     }
```

References _Ready(), and Instance.

Referenced by Hud.OnSaveMenuButtonPressed().

8.15.2.15 SetBVCurrentUses()

```
void PlayerStats.SetBVCurrentUses (
            int Value)  [inline]
```

Setzt die CurrentUses der Bloodvials.

Parameters

Value	Die CurrentUses der Bloodvials.

Definition at line 222 of file PlayerStats.cs.

```
00222                                        {
00223         BVCurrentUses = Math.Max(0, Value);
00224     }
```

References BVCurrentUses.

Referenced by StorageManager.LoadGameFile(), BloodVial.ResetUses(), and BloodVial.UseBloodVial().

8.15.2.16 SetBVHealAmount()

```
void PlayerStats.SetBVHealAmount (
            int Value)  [inline]
```

Setzt den HealAmount eines Bloodvials.

Parameters

Value	Den neuen Wert für den HealAmount.

Definition at line 190 of file PlayerStats.cs.

```
00190                                           {
00191         BVHealAmount = Math.Max(0, Value);
00192     }
```

References BVHealAmount.

Referenced by BloodVial.LevelHealAmount(), and StorageManager.LoadGameFile().

8.15.2.17 SetBVMaxUses()

```
void PlayerStats.SetBVMaxUses (
            int Value)  [inline]
```

Setzt die MaxUses der Bloodvials.

Parameters

Value	Die MaxUses der Bloodvials.

Definition at line 206 of file PlayerStats.cs.

```
00206                                         {
00207         BVMaxUses = Math.Max(0, Value);
00208     }
```

References BVMaxUses.

Referenced by BloodVial.AddMaxUses(), and StorageManager.LoadGameFile().

8.15.2.18 SetCurrentHealth()

```
void PlayerStats.SetCurrentHealth (
            float Health)  [inline]
```

Setzt die aktuellen Lebenspunkte des Spielers.

Parameters

Health	Neue Lebenspunkte, die gesetzt werden sollen.

Definition at line 147 of file PlayerStats.cs.

```
00147                                           {
00148         // CurrentHealth darf MaxHealthPoints nicht überschreiten.
00149         CurrentHealth = Mathf.Min(Health, MaxHealthPoints);
00150     }
```

References CurrentHealth, and MaxHealthPoints.

Referenced by StorageManager.LoadGameFile(), Player.MaxHeal(), Player.TakeDamage(), and BloodVial.UseBloodVial().

8.15.2.19 SetCurrentLevelTag()

```
void PlayerStats.SetCurrentLevelTag (
            String levelTag)  [inline]
```

Setter für CurrentLevelTag.

Parameters

CurrentLevelTag	Neuer Tag

Definition at line 64 of file PlayerStats.cs.

```
00064                                                   {
00065         CurrentLevelTag = levelTag;
00066     }
```

References CurrentLevelTag.

Referenced by NavigationManager.GoToLevel(), and StorageManager.LoadGameFile().

8.15.2.20 SetMaxHealthPoints()

```
void PlayerStats.SetMaxHealthPoints (
            float maxHealthPoints)  [inline]
```

Setzt die maximalen Lebenspunkte des Spielers.

Parameters

maxHealthPoints	Die neuen maximalen Lebenspunkte (muss positiv sein).

Definition at line 130 of file PlayerStats.cs.

```
00130                                                       {
00131         // MaxHealthPoints muss immer positiv sein
00132         MaxHealthPoints = Mathf.Max(maxHealthPoints, 1); // Verhindert, dass MaxHealthPoints <= 0 wird
00133     }
```

References MaxHealthPoints.

Referenced by StorageManager.LoadGameFile().

8.15.2.21 SetMaxStamina()

```
void PlayerStats.SetMaxStamina (
            float Value)  [inline]
```

Setzt die maximale Stamina des Spielers.

Parameters

Value	Den neuen Wert für die maximale Stamina (muss positiv sein).

Definition at line 156 of file PlayerStats.cs.

```
00156                                               {
00157         // MaxStamina muss immer positiv sein
00158         MaxStamina = Mathf.Max(Value, 1);
00159     }
```

References MaxStamina.

Referenced by StorageManager.LoadGameFile().

8.15.2.22 SetPosition()

```
void PlayerStats.SetPosition (
            Vector2 position)  [inline]
```

Setzt die Position des Spielers.

Parameters

Position	des Spielers.

Definition at line 88 of file PlayerStats.cs.

```
00088                                                {
00089         Position = position;
00090     }
```

References Position.

Referenced by Player._PhysicsProcess(), and StorageManager.LoadGameFile().

8.15.2.23 SetRespawnLevelTag()

```
void PlayerStats.SetRespawnLevelTag (
            String levelTag)  [inline]
```

Setter für RespawnLevelTag.

Parameters

RespawnLevelTag	Neuer Tag

Definition at line 48 of file PlayerStats.cs.

```
00048                                                    {
00049         RespawnLevelTag = levelTag;
00050     }
```

References RespawnLevelTag.

Referenced by StorageManager.LoadGameFile(), and Checkpoint.OnPlayerBodyEntered().

8.15.2.24 SetSinAmount()

```
void PlayerStats.SetSinAmount (
            int Value)  [inline]
```

Setzt den SinAmount des Spielers.

Parameters

Value	Der neue Wert für den SinAmount.

Definition at line 113 of file PlayerStats.cs.

```
00113                                          {
00114         // SinAmount muss immer >= 0 sein
00115         SinAmount = Mathf.Max(Value, 0);
00116     }
```

References SinAmount.

Referenced by StorageManager.LoadGameFile(), and Player.SetSinAmount().

8.15.2.25 SetSpawnPoint()

```
void PlayerStats.SetSpawnPoint (
            Vector2 spawnPoint)  [inline]
```

Setzt den SpawnPoint des Spielers.

Parameters

Der	SpawnPoint des Spielers.

Definition at line 72 of file PlayerStats.cs.

```
00072                                                {
00073         SpawnPoint = spawnPoint;
00074     }
```

References SpawnPoint.

Referenced by StorageManager.LoadGameFile(), and Checkpoint.OnPlayerBodyEntered().

8.15.2.26 SetStamina()

```
void PlayerStats.SetStamina (
             float Value)  [inline]
```

Setzt die Stamina des Spielers.

Parameters

Value	Den neuen Wert für Stamina (muss im Bereich zwischen 0 und MaxStamina liegen).

Definition at line 173 of file PlayerStats.cs.

```
00173                                          {
00174         // Stellt sicher, dass die CurrentStamina im gültigen Bereich bleibt (zwischen 0 und
      MaxStamina)
00175         CurrentStamina = Mathf.Clamp(Value, 0, MaxStamina);
00176     }
```

References CurrentStamina, and MaxStamina.

Referenced by StorageManager.LoadGameFile(), Checkpoint.OnPlayerBodyEntered(), Player.RegenerateStamina(), Player.TakeDamage(), and Player.UseStamina().

8.15.3 Member Data Documentation

8.15.3.1 BVCurrentUses

```
int PlayerStats.BVCurrentUses  [private]
```

Definition at line 23 of file PlayerStats.cs.

Referenced by _Ready(), GetBVCurrentUses(), and SetBVCurrentUses().

8.15.3.2 BVHealAmount

```
int PlayerStats.BVHealAmount = 25  [private]
```

Definition at line 21 of file PlayerStats.cs.

Referenced by GetBVHealAmount(), and SetBVHealAmount().

8.15.3.3 BVMaxUses

```
int PlayerStats.BVMaxUses = 5  [private]
```

Definition at line 22 of file PlayerStats.cs.

Referenced by _Ready(), GetBVMaxUses(), and SetBVMaxUses().

8.15.3.4 CurrentHealth

```
float PlayerStats.CurrentHealth  [private]
```

Definition at line 18 of file PlayerStats.cs.

Referenced by _Ready(), GetCurrentHealth(), and SetCurrentHealth().

8.15.3.5 CurrentLevelTag

```
String PlayerStats.CurrentLevelTag = "intro"  [private]
```

Definition at line 13 of file PlayerStats.cs.

Referenced by GetCurrentLevelTag(), and SetCurrentLevelTag().

8.15.3.6 CurrentStamina

```
float PlayerStats.CurrentStamina  [private]
```

Definition at line 20 of file PlayerStats.cs.

Referenced by _Ready(), GetStamina(), and SetStamina().

8.15.3.7 MaxHealthPoints

```
float PlayerStats.MaxHealthPoints = 100f  [private]
```

Definition at line 17 of file PlayerStats.cs.

Referenced by _Ready(), GetMaxHealthPoints(), SetCurrentHealth(), and SetMaxHealthPoints().

8.15.3.8 MaxStamina

```
float PlayerStats.MaxStamina = 100f  [private]
```

Definition at line 19 of file PlayerStats.cs.

Referenced by _Ready(), GetMaxStamina(), SetMaxStamina(), and SetStamina().

8.15.3.9 Position

```
Vector2 PlayerStats.Position = new Vector2(-540, 160)  [private]
```

Definition at line 15 of file PlayerStats.cs.

Referenced by GetPosition(), and SetPosition().

8.15.3.10 RespawnLevelTag

```
String PlayerStats.RespawnLevelTag = "intro"  [private]
```

Definition at line 12 of file PlayerStats.cs.

Referenced by GetRespawnLevelTag(), and SetRespawnLevelTag().

8.15.3.11 SinAmount

```
int PlayerStats.SinAmount  [private]
```

Definition at line 16 of file PlayerStats.cs.

Referenced by GetSinAmount(), and SetSinAmount().

8.15.3.12 SpawnPoint

```
Vector2 PlayerStats.SpawnPoint  [private]
```

Definition at line 14 of file PlayerStats.cs.

Referenced by GetSpawnPoint(), and SetSpawnPoint().

8.15.4 Property Documentation

8.15.4.1 Instance

```
PlayerStats PlayerStats.Instance  [static], [get], [private set]
```

Definition at line 10 of file PlayerStats.cs.

```
00010 { get; private set; }
```

Referenced by Player._PhysicsProcess(), HealthBar._Process(), StaminaBar._Process(), BloodVial._Ready(), HealthBar._Ready(), LevelManager._Ready(), Player._Ready(), _Ready(), StaminaBar._Ready(), BloodVial.AddMaxUses(), BaseEnemy.Die(), NavigationManager.GoToLevel(), BloodVial.LevelHealAmount(), StorageManager.LoadGameFile(), Player.MaxHeal(), MainMenu.OnContinueButtonPressed(), Checkpoint.OnPlayerBodyEntered(), MainMenu.OnSave1SelectPressed(), MainMenu.OnSave2SelectPressed(), MainMenu.OnSave3SelectPressed(), Hud.OnSaveMenuButtonPressed(), Player.RegenerateStamina(), Reload(), BloodVial.ResetUses(), Player.Respawn(), StorageManager.SaveGameFile(), Player.SetSinAmount(), Player.TakeDamage(), BloodVial.UseBloodVial(), and Player.UseStamina().

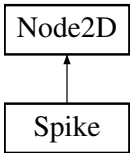
The documentation for this class was generated from the following file:

- C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/PlayerStats.cs

8.16 Spike Class Reference

Klasse für die Spikes.

Inheritance diagram for Spike:



Public Member Functions

- override void _Ready ()
  Initialisierung der Node Player.
- Damage GetDamage ()
  Gibt ein Damage Objekt zurück.

Private Member Functions

- void OnPlayerBodyEntered (Node body)
  Prüfen ob der Körper den Spike betritt falls ja wird der Timer gestartet und der Spieler nimmt Schaden.
- void OnPlayerBodyExited (Node body)
  Prüfen ob der Körper den Spike verlässt, falls ja wird der Timer gestoppt und der Spieler nimmt keinen Schaden mehr.
- void OnTimerTimeout ()
  Timer Timeout Methode, die den Schaden an den Spieler übergibt.

Private Attributes

- Player Player
- float Damage = 10f

8.16.1 Detailed Description

Klasse für die Spikes.

Definition at line 7 of file Spike.cs.

8.16.2 Member Function Documentation

8.16.2.1 _Ready()

```
override void Spike._Ready ()  [inline]
```

Initialisierung der Node Player.

Hier wird der Player Node gefunden

Definition at line 20 of file Spike.cs.

```
00021     {
00022         // Zugriff auf Player Node
00023
00024         Player = GetNode<Player>("../../Player");
00025     }
```

8.16.2.2 GetDamage()

```
Damage Spike.GetDamage ()  [inline]
```

Gibt ein Damage Objekt zurück.

Returns

Damage Objekt

Definition at line 71 of file Spike.cs.

```
00072     {
00073         return new Damage(0, Damage, Vector2.Zero, this);
00074     }
```

References Damage.

Referenced by OnPlayerBodyEntered(), and OnTimerTimeout().

8.16.2.3 OnPlayerBodyEntered()

```
void Spike.OnPlayerBodyEntered (
             Node body)  [inline], [private]
```

Prüfen ob der Körper den Spike betritt falls ja wird der Timer gestartet und der Spieler nimmt Schaden.

Definition at line 30 of file Spike.cs.

```
00031     {
00032
00033         if (body is Player)
00034         {
00035             Player = (Player)body; // Instanzvariable setzen
00036             Player.TakeDamage(GetDamage());
00037             Player.SlowPlayer(0.5f);
00038             GetNode<Timer>("StaticBody2D/Area2D/Timer").Start();
00039             GD.Print("Player entered spike");
00040         }
00041
00042
00043     }
```

References GetDamage(), Player, Player.SlowPlayer(), and Player.TakeDamage().

8.16.2.4 OnPlayerBodyExited()

```
void Spike.OnPlayerBodyExited (
             Node body)  [inline], [private]
```

Prüfen ob der Körper den Spike verlässt, falls ja wird der Timer gestoppt und der Spieler nimmt keinen Schaden mehr.

Definition at line 48 of file Spike.cs.

```
00049     {
00050         if (body is Player)
00051         {
00052             Player = null; // Instanzvariable zurücksetzen
00053             GetNode<Timer>("StaticBody2D/Area2D/Timer").Stop();
00054         }
00055     }
```

8.16.2.5 OnTimerTimeout()

```
void Spike.OnTimerTimeout ()  [inline], [private]
```

Timer Timeout Methode, die den Schaden an den Spieler übergibt.

Definition at line 60 of file Spike.cs.

```
00061     {
00062         GD.Print("Timer timeout");
00063         Player.TakeDamage(GetDamage());
00064         GetNode<Timer>("StaticBody2D/Area2D/Timer").Start();
00065     }
```

References GetDamage(), and Player.TakeDamage().

8.16.3 Member Data Documentation

8.16.3.1 Damage

```
float Spike.Damage = 10f  [private]
```

Definition at line 14 of file Spike.cs.

Referenced by GetDamage().

8.16.3.2 Player

```
Player Spike.Player  [private]
```

Definition at line 10 of file Spike.cs.

Referenced by OnPlayerBodyEntered().

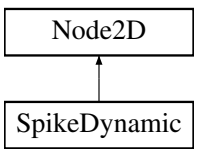
The documentation for this class was generated from the following file:

- C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Spike.cs

8.17 SpikeDynamic Class Reference

Klasse für die beweglichen Spikes.

Inheritance diagram for SpikeDynamic:



Public Member Functions

- override void _Ready ()

  Initialisierung der Node Player.
- Damage GetDamage ()

  Gibt ein Damage Objekt zurück.

Private Member Functions

- void OnPlayerBodyEntered (Node body)

  Prüfen ob der Körper den Spike betritt falls ja wird der Timer gestartet und der Spieler nimmt Schaden.
- void OnPlayerBodyExited (Node body)

  Prüfen ob der Körper den Spike verlässt, falls ja wird der Timer gestoppt und der Spieler nimmt keinen Schaden mehr.
- void OnTimerTimeout ()

  Timer Timeout Methode, die den Schaden an den Spieler übergibt.

Private Attributes

- Player Player
- float Damage = 10f

8.17.1 Detailed Description

Klasse für die beweglichen Spikes.

Definition at line 7 of file SpikeDynamic.cs.

8.17.2 Member Function Documentation

8.17.2.1 _Ready()

```
override void SpikeDynamic._Ready ()  [inline]
```

Initialisierung der Node Player.

Hier wird der Player Node gefunden

Definition at line 20 of file SpikeDynamic.cs.

```
00021     {
00022         // Zugriff auf Player Node
00023
00024         Player = GetNode<Player>("../../../Player");
00025     }
```

8.17.2.2 GetDamage()

```
Damage SpikeDynamic.GetDamage ()  [inline]
```

Gibt ein Damage Objekt zurück.

Returns

Damage Objekt

Definition at line 71 of file SpikeDynamic.cs.

```
00072     {
00073         return new Damage(0, Damage, Vector2.Zero, this);
00074     }
```

References Damage.

Referenced by OnPlayerBodyEntered(), and OnTimerTimeout().

8.17.2.3 OnPlayerBodyEntered()

```
void SpikeDynamic.OnPlayerBodyEntered (
             Node body)  [inline], [private]
```

Prüfen ob der Körper den Spike betritt falls ja wird der Timer gestartet und der Spieler nimmt Schaden.

Definition at line 30 of file SpikeDynamic.cs.

```
00031     {
00032
00033         if (body is Player)
00034         {
00035             Player = (Player)body; // Instanzvariable setzen
00036             Player.TakeDamage(GetDamage());
00037             Player.SlowPlayer(0.5f);
00038             GetNode<Timer>("StaticBody2D/Area2D/Timer").Start();
00039             GD.Print("Player entered spike");
00040         }
00041
00042
00043     }
```

References GetDamage(), Player, Player.SlowPlayer(), and Player.TakeDamage().

8.17.2.4 OnPlayerBodyExited()

```
void SpikeDynamic.OnPlayerBodyExited (
             Node body)  [inline], [private]
```

Prüfen ob der Körper den Spike verlässt, falls ja wird der Timer gestoppt und der Spieler nimmt keinen Schaden mehr.

Definition at line 48 of file SpikeDynamic.cs.

```
00049     {
00050         if (body is Player)
00051         {
00052             Player = null; // Instanzvariable zurücksetzen
00053             GetNode<Timer>("StaticBody2D/Area2D/Timer").Stop();
00054         }
00055     }
```

8.17.2.5 OnTimerTimeout()

```
void SpikeDynamic.OnTimerTimeout ()  [inline], [private]
```

Timer Timeout Methode, die den Schaden an den Spieler übergibt.

Definition at line 60 of file SpikeDynamic.cs.

```
00061     {
00062         GD.Print("Timer timeout");
00063         Player.TakeDamage(GetDamage());
00064         GetNode<Timer>("StaticBody2D/Area2D/Timer").Start();
00065     }
```

References GetDamage(), and Player.TakeDamage().

8.17.3 Member Data Documentation

8.17.3.1 Damage

```
float SpikeDynamic.Damage = 10f  [private]
```

Definition at line 13 of file SpikeDynamic.cs.

Referenced by GetDamage().

8.17.3.2 Player

```
Player SpikeDynamic.Player  [private]
```

Definition at line 10 of file SpikeDynamic.cs.

Referenced by OnPlayerBodyEntered().

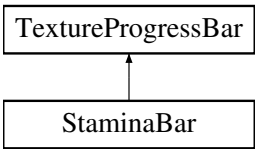
The documentation for this class was generated from the following file:

- C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/SpikeDynamic.cs

8.18 StaminaBar Class Reference

Klasse für die Ausdauerleiste des Spielers. Synchronisiert die Anzeige der StaminaBar mit der Ausdauer des Spielers.

Inheritance diagram for StaminaBar:



Public Member Functions

- override void _Ready ()

  Initialisiert die StaminaBar und verbindet sie mit der Ausdauer des Spielers. Lädt den Spieler-Knoten und setzt die maximale und aktuelle Ausdauer in der StaminaBar.
- override void _Process (double DeltaTime)

  Aktualisiert die StaminaBar in jedem Frame. Synchronisiert die Anzeige der aktuellen Ausdauer mit den Werten des Spielers.

8.18.1 Detailed Description

Klasse für die Ausdauerleiste des Spielers. Synchronisiert die Anzeige der StaminaBar mit der Ausdauer des Spielers.

Definition at line 7 of file StaminaBar.cs.

8.18.2 Member Function Documentation

8.18.2.1 _Process()

```
override void StaminaBar._Process (
            double DeltaTime)  [inline]
```

Aktualisiert die StaminaBar in jedem Frame. Synchronisiert die Anzeige der aktuellen Ausdauer mit den Werten des Spielers.

Parameters

delta	Zeit seit dem letzten Frame (wird nicht direkt genutzt).

Definition at line 24 of file StaminaBar.cs.

```
00024                                           {
00025         // Aktualisiere den Wert der StaminaBar basierend auf der aktuellen Ausdauer des Spielers
00026         Value = PlayerStats.Instance.GetStamina();
00027     }
```

References PlayerStats.GetStamina(), and PlayerStats.Instance.

8.18.2.2 _Ready()

```
override void StaminaBar._Ready ()  [inline]
```

Initialisiert die StaminaBar und verbindet sie mit der Ausdauer des Spielers. Lädt den Spieler-Knoten und setzt die maximale und aktuelle Ausdauer in der StaminaBar.

Definition at line 13 of file StaminaBar.cs.

```
00013                                {
00014         // Setze die maximale Ausdauer der StaminaBar basierend auf der Spieler-MaxStamina
00015         MaxValue = PlayerStats.Instance.GetMaxStamina();
00016         Value = PlayerStats.Instance.GetStamina();
00017     }
```

References PlayerStats.GetMaxStamina(), PlayerStats.GetStamina(), and PlayerStats.Instance.

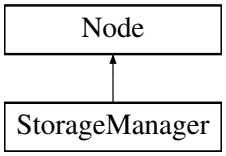
The documentation for this class was generated from the following file:

- C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/StaminaBar.cs

8.19 StorageManager Class Reference

Klasse für das Speichern und Laden von Daten.

Inheritance diagram for StorageManager:



Public Member Functions

- override void _Ready ()

  Initialisierung der Instanz und erstes laden der Einstellungen.
- void LoadSettings ()

  Laden der Einstellungen.
- void LoadGameFile (int id)

  Laden eines Spielstandes.
- void SaveAll (int id)

  Speichern der Einstellungen und eines Spielstandes.
- void SaveSettings ()

  Speichern der Einstellungen.
- void SaveGameFile (int id)

  Speichern eines Spielstandes.
- void SetLastSaveId (int id)

  Setter für LastSaveId.
- int GetLastSaveId ()

  Getter für LastSaveId.
- void SetSaves (int Saves)

  Setter für Saves.
- int GetSaves ()

  Getter für Saves.

Properties

- static StorageManager Instance `[get, private set]`

Private Attributes

- String[ ] PathSave = {"user://save1.dat", "user://save2.dat", "user://save3.dat"}
- int LastSaveId = -1
- int Saves = 0

Static Private Attributes

- const String PathSettings = "user://settings.txt"

8.19.1 Detailed Description

Klasse für das Speichern und Laden von Daten.

Definition at line 8 of file StorageManager.cs.

8.19.2 Member Function Documentation

8.19.2.1 _Ready()

```
override void StorageManager._Ready ()  [inline]
```

Initialisierung der Instanz und erstes laden der Einstellungen.

Definition at line 20 of file StorageManager.cs.

```
00020                                {
00021         LoadSettings();
00022         Instance = this;
00023     }
```

References Instance, and LoadSettings().

8.19.2.2 GetLastSaveId()

```
int StorageManager.GetLastSaveId ()  [inline]
```

Getter für LastSaveId.

Returns

int LastSaveId

Definition at line 118 of file StorageManager.cs.

```
00118                               {
00119         return LastSaveId;
00120     }
```

References LastSaveId.

Referenced by MainMenu._Ready(), MainMenu.OnContinueButtonPressed(), Hud.OnSaveButtonPressed(), Hud.OnSaveMenuButtonPressed(), and Hud.OnSaveQuitButtonPressed().

8.19.2.3 GetSaves()

```
int StorageManager.GetSaves ()  [inline]
```

Getter für Saves.

Returns

int Saves

Definition at line 134 of file StorageManager.cs.

```
00134                           {
00135         return Saves;
00136     }
```

References Saves.

Referenced by MainMenu.Change(), MainMenu.OnDeleteConfirmationConfirmed(), MainMenu.OnSave1SelectPressed(), MainMenu.OnSave2SelectPressed(), and MainMenu.OnSave3SelectPressed().

8.19.2.4 LoadGameFile()

```
void StorageManager.LoadGameFile (
            int id)  [inline]
```

Laden eines Spielstandes.

Parameters

Id	des Spielstandes.

Definition at line 43 of file StorageManager.cs.

```
00043                                       {
00044         if(!FileAccess.FileExists(PathSave[id])){
00045             return;
00046         }
00047         FileAccess File = FileAccess.Open(PathSave[id], FileAccess.ModeFlags.Read);
00048         PlayerStats.Instance.SetRespawnLevelTag((String) File.GetVar());
00049         PlayerStats.Instance.SetCurrentLevelTag((String) File.GetVar());
00050         PlayerStats.Instance.SetSpawnPoint((Vector2) File.GetVar());
00051         PlayerStats.Instance.SetPosition((Vector2) File.GetVar());
00052         PlayerStats.Instance.SetSinAmount((int) File.GetVar());
00053         PlayerStats.Instance.SetMaxHealthPoints((float) File.GetVar());
00054         PlayerStats.Instance.SetCurrentHealth((float) File.GetVar());
00055         PlayerStats.Instance.SetMaxStamina((float) File.GetVar());
00056         PlayerStats.Instance.SetStamina((float) File.GetVar());
00057         PlayerStats.Instance.SetBVHealAmount((int) File.GetVar());
00058         PlayerStats.Instance.SetBVMaxUses((int) File.GetVar());
00059         PlayerStats.Instance.SetBVCurrentUses((int) File.GetVar());
00060
00061         File.Close();
00062     }
```

References PlayerStats.Instance, PathSave, PlayerStats.SetBVCurrentUses(), PlayerStats.SetBVHealAmount(), PlayerStats.SetBVMaxUses(), PlayerStats.SetCurrentHealth(), PlayerStats.SetCurrentLevelTag(), PlayerStats.SetMaxHealthPoints(), PlayerStats.SetMaxStamina(), PlayerStats.SetPosition(), PlayerStats.SetRespawnLevelTag(), PlayerStats.SetSinAmount(), PlayerStats.SetSpawnPoint(), and PlayerStats.SetStamina().

Referenced by MainMenu.OnContinueButtonPressed(), MainMenu.OnSave1SelectPressed(), MainMenu.OnSave2SelectPressed(), and MainMenu.OnSave3SelectPressed().

8.19.2.5 LoadSettings()

```
void StorageManager.LoadSettings ()  [inline]
```

Laden der Einstellungen.

Definition at line 28 of file StorageManager.cs.

```
00028                                {
00029         if(!FileAccess.FileExists(PathSettings)){
00030             return;
00031         }
00032         FileAccess File = FileAccess.Open(PathSettings, FileAccess.ModeFlags.Read);
00033         Saves = (int) File.GetVar();
00034         LastSaveId = (int) File.GetVar();
00035
00036         File.Close();
00037     }
```

References LastSaveId, PathSettings, and Saves.

Referenced by _Ready().

8.19.2.6 SaveAll()

```
void StorageManager.SaveAll (
            int id)  [inline]
```

Speichern der Einstellungen und eines Spielstandes.

Parameters

Id	des Spielstandes.

Definition at line 68 of file StorageManager.cs.

```
00068                                   {
00069         SaveGameFile(id);
00070         SaveSettings();
00071     }
```

References SaveGameFile(), and SaveSettings().

Referenced by Hud.OnSaveButtonPressed(), Hud.OnSaveMenuButtonPressed(), and Hud.OnSaveQuitButtonPressed().

8.19.2.7 SaveGameFile()

```
void StorageManager.SaveGameFile (
            int id)  [inline]
```

Speichern eines Spielstandes.

Parameters

Id	des Spielstandes.

Definition at line 88 of file StorageManager.cs.

```
00088                                   {
00089         FileAccess File = FileAccess.Open(PathSave[id], FileAccess.ModeFlags.Write);
00090         File.StoreVar(PlayerStats.Instance.GetRespawnLevelTag());
00091         File.StoreVar(PlayerStats.Instance.GetCurrentLevelTag());
00092         File.StoreVar(PlayerStats.Instance.GetSpawnPoint());
00093         File.StoreVar(PlayerStats.Instance.GetPosition());
00094         File.StoreVar(PlayerStats.Instance.GetSinAmount());
00095         File.StoreVar(PlayerStats.Instance.GetMaxHealthPoints());
00096         File.StoreVar(PlayerStats.Instance.GetCurrentHealth());
00097         File.StoreVar(PlayerStats.Instance.GetMaxStamina());
00098         File.StoreVar(PlayerStats.Instance.GetStamina());
00099         File.StoreVar(PlayerStats.Instance.GetBVHealAmount());
00100         File.StoreVar(PlayerStats.Instance.GetBVMaxUses());
00101         File.StoreVar(PlayerStats.Instance.GetBVCurrentUses());
00102
00103         File.Close();
00104     }
```

References PlayerStats.GetBVCurrentUses(), PlayerStats.GetBVHealAmount(), PlayerStats.GetBVMaxUses(), PlayerStats.GetCurrentHealth(), PlayerStats.GetCurrentLevelTag(), PlayerStats.GetMaxHealthPoints(), PlayerStats.GetMaxStamina(), PlayerStats.GetPosition(), PlayerStats.GetRespawnLevelTag(), PlayerStats.GetSinAmount(), PlayerStats.GetSpawnPoint(), PlayerStats.GetStamina(), PlayerStats.Instance, and PathSave.

Referenced by SaveAll().

8.19.2.8 SaveSettings()

```
void StorageManager.SaveSettings ()  [inline]
```

Speichern der Einstellungen.

Definition at line 76 of file StorageManager.cs.

```
00076                                   {
00077         FileAccess File = FileAccess.Open(PathSettings, FileAccess.ModeFlags.Write);
00078         File.StoreVar(Saves);
00079         File.StoreVar(LastSaveId);
00080
00081         File.Close();
00082     }
```

References LastSaveId, PathSettings, and Saves.

Referenced by MainMenu.OnQuitButtonPressed(), and SaveAll().

8.19.2.9 SetLastSaveId()

```
void StorageManager.SetLastSaveId (
            int id)  [inline]
```

Setter für LastSaveId.

Parameters

int	LastSaveId

Definition at line 110 of file StorageManager.cs.

```
00110                                       {
00111         LastSaveId = id;
00112     }
```

References LastSaveId.

Referenced by MainMenu.OnSave1SelectPressed(), MainMenu.OnSave2SelectPressed(), and MainMenu.OnSave3SelectPressed().

8.19.2.10 SetSaves()

```
void StorageManager.SetSaves (
            int Saves)  [inline]
```

Setter für Saves.

Parameters

int	Saves

Definition at line 126 of file StorageManager.cs.

```
00126                                     {
00127         this.Saves = Saves;
00128     }
```

References Saves.

Referenced by MainMenu.OnDeleteConfirmationConfirmed(), MainMenu.OnSave1SelectPressed(), MainMenu.OnSave2SelectPresse
and MainMenu.OnSave3SelectPressed().

8.19.3 Member Data Documentation

8.19.3.1 LastSaveId

```
int StorageManager.LastSaveId = -1  [private]
```

Definition at line 13 of file StorageManager.cs.

Referenced by GetLastSaveId(), LoadSettings(), SaveSettings(), and SetLastSaveId().

8.19.3.2 PathSave

```
String [] StorageManager.PathSave = {"user://save1.dat", "user://save2.dat", "user://save3.↩
dat"}  [private]
```

Definition at line 12 of file StorageManager.cs.
```
00012 {"user://save1.dat", "user://save2.dat", "user://save3.dat"};
```

Referenced by LoadGameFile(), and SaveGameFile().

8.19.3.3 PathSettings

```
const String StorageManager.PathSettings = "user://settings.txt"  [static], [private]
```

Definition at line 11 of file StorageManager.cs.

Referenced by LoadSettings(), and SaveSettings().

8.19.3.4 Saves

```
int StorageManager.Saves = 0  [private]
```

Definition at line 14 of file StorageManager.cs.

Referenced by GetSaves(), LoadSettings(), SaveSettings(), and SetSaves().

8.19.4 Property Documentation

8.19.4.1 Instance

```
StorageManager StorageManager.Instance  [static], [get], [private set]
```

Definition at line 10 of file StorageManager.cs.
```
00010 { get; private set; }
```

Referenced by MainMenu._Ready(), _Ready(), MainMenu.Change(), MainMenu.OnContinueButtonPressed(), MainMenu.OnDeleteConfirmationConfirmed(), MainMenu.OnQuitButtonPressed(), MainMenu.OnSave1SelectPressed(), MainMenu.OnSave2SelectPressed(), MainMenu.OnSave3SelectPressed(), Hud.OnSaveButtonPressed(), Hud.OnSaveMenuButtonPressed(), and Hud.OnSaveQuitButtonPressed().

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

- C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/StorageManager.cs

Chapter 9

File Documentation

9.1 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/addons/GDMUT/Dock.cs File Reference

Namespaces

- namespace GdMUT
- namespace GdMUT.Components

9.2 Dock.cs

Go to the documentation of this file.

```
00001 // Copyright (c) Spencer (Spycemyster) Chang, LLC. All Rights Reserved.
00002 // Licensed under the MIT License. See LICENSE in the project root for license information.
00003 namespace GdMUT.Components;
00004
00005 using Godot;
00006 using System;
00007 using System.Diagnostics;
00008 using System.Threading;
00009
00010 #if TOOLS
00014 [Tool]
00015 public partial class Dock : Control
00016 {
00017     private const string TEST_RESULT_SCENE = "res://addons/GDMUT/TestResult.tscn";
00018     private readonly System.Collections.Generic.Dictionary<
00019         Type,
00020         System.Collections.Generic.List<TestFunction>
00021     > _testDictionary = new();
00022     private readonly System.Collections.Generic.Dictionary<Type, TestResult> _testResultDictionary =
00023         new();
00024
00025     [Export]
00026     private LineEdit _filter;
00027
00028     [Export]
00029     private CheckBox _multithreadedEnabled;
00030
00031     [Export]
00032     private LineEdit _numThreads;
00033
00034     [Export]
00035     private Button _runTests;
00036
00037     [Export]
00038     private Button _loadTests;
00039
00040     [Export]
00041     private VBoxContainer _testList;
```

```
00042
00043     private System.Collections.Generic.List<TestFunction> _tests = new();
00044
00048     public override void _EnterTree()
00049     {
00050         base._EnterTree();
00051         _runTests.Pressed += RunTests;
00052         _loadTests.Pressed += LoadTests;
00053     }
00054
00055     private void LoadTests()
00056     {
00057         var stopwatch = new Stopwatch();
00058         stopwatch.Start();
00059         foreach (Node node in _testList.GetChildren())
00060         {
00061             node.QueueFree();
00062         }
00063
00064         _tests?.Clear();
00065         _tests = TestLoader.SearchForAllTests();
00066         _testDictionary.Clear();
00067         for (int testIndex = 0; testIndex < _tests.Count; testIndex++)
00068         {
00069             TestFunction function = _tests[testIndex];
00070             if (!function.Name.Contains(_filter.Text))
00071             {
00072                 continue;
00073             }
00074
00075             if (
00076                 _testDictionary.TryGetValue(
00077                     function.Type,
00078                     out System.Collections.Generic.List<TestFunction> testList
00079                 )
00080             )
00081             {
00082                 testList.Add(function);
00083             }
00084             else
00085             {
00086                 _testDictionary.Add(
00087                     function.Type,
00088                     new System.Collections.Generic.List<TestFunction>() { function }
00089                 );
00090             }
00091         }
00092
00093         _testResultDictionary.Clear();
00094         var testResultScene = GD.Load<PackedScene>(TEST_RESULT_SCENE);
00095         foreach (Type type in _testDictionary.Keys)
00096         {
00097             var functions = _testDictionary[type];
00098             var testResult = testResultScene.Instantiate<TestResult>();
00099             testResult.SetTypeName(type.Name);
00100             _testList.AddChild(testResult);
00101             _testResultDictionary.Add(type, testResult);
00102             foreach (TestFunction function in functions)
00103             {
00104                 testResult.AddMethodResult(function);
00105             }
00106         }
00107
00108         stopwatch.Stop();
00109         GD.Print($"Loading tests took {stopwatch.ElapsedMilliseconds}ms");
00110     }
00111
00112     private void RunTestsInRange(int startIndex, int endIndex)
00113     {
00114         for (int testIndex = startIndex; testIndex < endIndex; testIndex++)
00115         {
00116             var test = _tests[testIndex];
00117             GD.Print(test.Name);
00118             Result testResult;
00119             try
00120             {
00121                 testResult = (Result)test.Method.Invoke(null, null);
00122             }
00123             catch (Exception e)
00124             {
00125                 testResult = new Result(false, $"Exception thrown: {e.Message}");
00126             }
00127
00128             test.Result = testResult;
00129         }
00130     }
00131
```

```
00132     private void RunTests()
00133     {
00134         if (_tests.Count == 0)
00135         {
00136             GD.Print("No tests loaded");
00137             return;
00138         }
00139 
00140         var stopwatch = new Stopwatch();
00141         stopwatch.Start();
00142 
00143         if (
00144             _multithreadedEnabled.ButtonPressed
00145             && int.TryParse(_numThreads.Text, out int numThreads)
00146             && numThreads > 0
00147         )
00148         {
00149             GD.Print("Run Tests multithreaded");
00150             Thread[] threads = new Thread[numThreads];
00151             int testsPerThread =
00152                 (_tests.Count / numThreads) + (_tests.Count % numThreads > 0 ? 1 : 0);
00153             for (int threadIndex = 0; threadIndex < numThreads; threadIndex++)
00154             {
00155                 int startIndex = threadIndex * testsPerThread;
00156                 int endIndex = Math.Min((threadIndex + 1) * testsPerThread, _tests.Count);
00157                 threads[threadIndex] = new Thread(() => RunTestsInRange(startIndex, endIndex));
00158                 threads[threadIndex].Start();
00159             }
00160 
00161             foreach (Thread thread in threads)
00162             {
00163                 thread.Join();
00164             }
00165         }
00166         else
00167         {
00168             GD.Print("Run Tests singlethreaded");
00169             RunTestsInRange(0, _tests.Count);
00170         }
00171 
00172         stopwatch.Stop();
00173         UpdateUIWithResults();
00174         GD.Print($"Tests took {stopwatch.ElapsedMilliseconds}ms");
00175     }
00176 
00177     private void UpdateUIWithResults()
00178     {
00179         foreach (TestResult result in _testResultDictionary.Values)
00180         {
00181             result.UpdateResult();
00182         }
00183     }
00184 }
00185 #endif
```

9.3 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/addons/GDMUT/GDMUT.cs File Reference

9.4 GDMUT.cs

Go to the documentation of this file.

```
00001 #if TOOLS
00002 using Godot;
00003 
00004 namespace GdMUT;
00005 
00009 [Tool]
00010 public partial class GDMUT : EditorPlugin
00011 {
00012     private const string DOCK_SCENE = "res://addons/GDMUT/Dock.tscn";
00013     private Control _dock;
00014 
00018     public override void _EnterTree()
00019     {
00020         base._EnterTree();
00021         _dock = GD.Load<PackedScene>(DOCK_SCENE).Instantiate<Control>();
00022         AddControlToDock(DockSlot.RightUl, _dock);
```

```
00023         GD.Print("Successfully loaded GDMUT");
00024     }
00025
00029     public override void _ExitTree()
00030     {
00031         base._ExitTree();
00032         RemoveControlFromDocks(_dock);
00033         _dock?.Free();
00034     }
00035 }
00036 #endif
```

9.5 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/addons/GDMUT/MethodResult.cs File Reference

9.6 MethodResult.cs

Go to the documentation of this file.

```
00001 #if TOOLS
00002 using Godot;
00003
00004 namespace GdMUT.Components;
00005
00009 [Tool]
00010 public partial class MethodResult : Control
00011 {
00012     [Export]
00013     private RichTextLabel _methodName;
00014
00015     [Export]
00016     private RichTextLabel _result;
00017     private TestFunction _function;
00018
00022     public override void _EnterTree()
00023     {
00024         base._EnterTree();
00025     }
00026
00031     public void SetMethodResult(TestFunction function)
00032     {
00033         _function = function;
00034         _methodName.Text = function.Method.Name;
00035         Reset();
00036     }
00037
00041     public void Update()
00042     {
00043         SetSuccess(_function.Result.IsSuccess, _function.Result.Message);
00044     }
00045
00049     public void Reset()
00050     {
00051         _result.Text = string.Empty;
00052         SelfModulate = new Color(1, 1, 1);
00053     }
00054
00060     public void SetSuccess(bool isSuccess, string result = "")
00061     {
00062         _result.Text = (isSuccess ? "Success: " : "Failure: ") + result;
00063         Modulate = isSuccess ? new Color(0, 1, 0) : new Color(1, 0, 0);
00064         GD.Print($"{result} {isSuccess}");
00065     }
00066 }
00067 #endif
```

9.7 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/addons/GDMUT/Result.cs File Reference

9.8 Result.cs

Go to the documentation of this file.

```
00001 #if TOOLS
00002 using System;
00003
00004 namespace GdMUT;
00005
00009 public struct Result
00010 {
00014     public static readonly Result Success = new(true, string.Empty);
00015
00019     public static readonly Result Failure = new(false, string.Empty);
00020
00026     public Result(bool success, string message = "")
00027     {
00028         IsSuccess = success;
00029         Message = message;
00030     }
00031
00035     public bool IsSuccess { get; set; }
00036
00040     public string Message { get; set; }
00041 }
00042 #endif
```

9.9 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/addons/GDMUT/TestFunction.cs File Reference

9.10 TestFunction.cs

Go to the documentation of this file.

```
00001 #if TOOLS
00002 using System;
00003 using System.Reflection;
00004
00005 namespace GdMUT;
00006
00010 public class TestFunction
00011 {
00015     public string Name { get; set; }
00016
00020     public Type Type { get; set; }
00021
00025     public MethodInfo Method { get; set; }
00026
00030     public Result Result { get; set; }
00031 }
00032 #endif
```

9.11 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/addons/GDMUT/TestLoader.cs File Reference

9.12 TestLoader.cs

Go to the documentation of this file.

```
00001 #if TOOLS
00002 using System;
00003 using System.Collections.Generic;
00004 using System.Reflection;
00005 using Godot;
00006
00007 namespace GdMUT;
00008
00012 public static class TestLoader
00013 {
00018     public static List<TestFunction> SearchForAllTests()
00019     {
00020         List<TestFunction> tests = new();
00021
00022         // get all functions with MonoTestFunctionAttribute
00023         ReadOnlySpan<Assembly> assemblies = AppDomain.CurrentDomain.GetAssemblies();
00024         for (int assemblyIndex = 0; assemblyIndex < assemblies.Length; assemblyIndex++)
00025         {
00026             Assembly assembly = assemblies[assemblyIndex];
00027             if (
00028                 assembly.FullName.StartsWith("System.")
00029                 || assembly.FullName.Equals("System")
00030                 || assembly.FullName.StartsWith("Microsoft.")
00031                 || assembly.FullName.StartsWith("GodotSharp")
00032                 || assembly.FullName.StartsWith("GodotTools")
00033                 || assembly.FullName.StartsWith("GodotPlugins")
00034                 || assembly.FullName.StartsWith("JetBrains")
00035                 || assembly.FullName.Equals("netstandard")
00036             )
00037             {
00038                 continue;
00039             }
00040
00041             GD.Print($"Loading tests from {assembly.FullName}");
00042             LoadFunctionsFromAssembly(tests, assembly);
00043         }
00044
00045         return tests;
00046     }
00047
00048     private static void LoadFunctionsFromAssembly(List<TestFunction> tests, Assembly assembly)
00049     {
00050         ReadOnlySpan<Type> types = assembly.GetTypes();
00051         for (int typeIndex = 0; typeIndex < types.Length; typeIndex++)
00052         {
00053             LoadFunctionsFromType(tests, types[typeIndex]);
00054         }
00055     }
00056
00057     private static void LoadFunctionsFromType(List<TestFunction> tests, Type type)
00058     {
00059         ReadOnlySpan<MethodInfo> methods = type.GetMethods();
00060         foreach (var method in methods)
00061         {
00062             var attribute = method.GetCustomAttributes(typeof(CSTestFunctionAttribute), false);
00063
00064             if (attribute.Length > 0)
00065             {
00066                 if (method.ReturnType != typeof(Result))
00067                 {
00068                     GD.PushError(
00069                         $"Method {method.Name} in {method.DeclaringType} does not return Result.
      Skipping it..."
00070                     );
00071                     continue;
00072                 }
00073                 else if (!method.IsStatic)
00074                 {
00075                     GD.PushError(
00076                         $"Method {method.Name} in {method.DeclaringType} is not static. Skipping
      it..."
00077                     );
00078                     continue;
00079                 }
00080
00081                 tests.Add(
00082                     new TestFunction()
00083                     {
00084                         Name = method.Name,
00085                         Type = method.DeclaringType,
00086                         Method = method,
00087                     }
00088                 );
00089             }
00090         }
00091     }
00092 }
```

```
00093
00097 [AttributeUsage(AttributeTargets.Method)]
00098 public class CSTestFunctionAttribute : Attribute
00099 {
00103     public CSTestFunctionAttribute() { }
00104 }
00105 #endif
```

9.13 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/addons/GDMUT/TestResult.cs File Reference

9.14 TestResult.cs

Go to the documentation of this file.

```
00001 #if TOOLS
00002 using System.Collections.Generic;
00003 using Godot;
00004
00005 namespace GdMUT.Components;
00006
00010 [Tool]
00011 public partial class TestResult : Control
00012 {
00013     private const string TYPE_NAME_FORMAT = "[b][font_size=24][center]{0}[/center][/font_size][/b]";
00014     private const string METHOD_RESULT_SCENE = "res://addons/GDMUT/MethodResult.tscn";
00015
00016     [Export]
00017     private RichTextLabel _typeName;
00018
00019     [Export]
00020     private VBoxContainer _methodList;
00021
00022     private List<(MethodResult, TestFunction)> _functions = new();
00023     private string _typeNameStr;
00024
00026     public override void _EnterTree()
00027     {
00028         base._EnterTree();
00029         foreach (Node child in _methodList.GetChildren())
00030         {
00031             child.QueueFree();
00032         }
00033
00034         _functions.Clear();
00035     }
00036
00041     public void SetTypeName(string typeName)
00042     {
00043         _typeNameStr = typeName;
00044         _typeName.Text = string.Format(TYPE_NAME_FORMAT, typeName);
00045     }
00046
00050     public void UpdateResult()
00051     {
00052         int numSuccess = 0;
00053         foreach (var (methodResult, function) in _functions)
00054         {
00055             methodResult.Update();
00056             numSuccess += function.Result.IsSuccess ? 1 : 0;
00057         }
00058
00059         _typeName.Text = string.Format(
00060             TYPE_NAME_FORMAT,
00061             _typeNameStr + $" ({numSuccess}/{_functions.Count})"
00062         );
00063         if (numSuccess == _functions.Count)
00064         {
00065             _typeName.Modulate = new Color(0, 1, 0);
00066         }
00067         else if (numSuccess == 0)
00068         {
00069             _typeName.Modulate = new Color(1, 0, 0);
00070         }
00071         else
00072         {
```

```
00073             _typeName.Modulate = new Color(1, 0.9f, 0);
00074         }
00075     }
00076
00081     public void AddMethodResult(TestFunction function)
00082     {
00083         var methodResultScene = GD.Load<PackedScene>(METHOD_RESULT_SCENE);
00084         var methodResult = methodResultScene.Instantiate<MethodResult>();
00085         methodResult.SetMethodResult(function);
00086         _methodList.AddChild(methodResult);
00087         _functions.Add((methodResult, function));
00088     }
00089 }
00090 #endif
```

9.15 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/addons/godot-git-plugin/↩ THIRDPARTY.md File Reference

9.16 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/README.md File Reference

9.17 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/scripts/BaseEnemy.cs File Reference

Classes

- class BaseEnemy

  Klasse für einen einfachen Gegner.

9.18 BaseEnemy.cs

Go to the documentation of this file.

```
00001 using Godot;
00002 using System;
00003
00007 public partial class BaseEnemy : CharacterBody2D
00008 {
00009
00010     private enum State {
00011         IDLE, WALK, ATTACK, TAKE_HIT
00012     }
00013
00014     //customizable variables
00015     [Export]
00016     protected float Damage = 20f;
00017     [Export]
00018     protected bool Dead = false;
00019     [Export]
00020     protected bool Respawnable = true;
00021     [Export]
00022     protected float MaxHealthPoints = 100f;
00023     [Export]
00024     protected float Armor = 20f; //MUSS ZISCHEN 0 UND 99 LIEGEN
00025     [Export]
00026     protected float MaxStamina = 1f;
00027     [Export]
00028     protected float Speed = 10;
00029     [Export]
00030     protected int SinAmount = 10;
00031     [Export]
00032     protected double ReturnToStartAfter = 5;
```

```
00033     [Export(PropertyHint.Flags,
      "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")]
00034     public uint Id { get; set;} = 0;
00035
00036     //private variables
00037     protected float CurrentHealthPoints;
00038     protected float CurrentStamina;
00039     protected double ReturnToStart;
00040     protected bool Pursuing = false;
00041     protected Node2D CurrentTarget = null;
00042     protected Vector2 TargetPosition = Vector2.Inf;
00043     protected Vector2 StartPosition;
00044     protected bool StartRotation = false;
00045     private State AnimationState = State.IDLE;
00046     protected bool AlreadyHit = false;
00047
00048     //linked nodes
00049     protected AnimatedSprite2D Sprite;
00050     protected CollisionPolygon2D CollisionPolygon;
00051     protected Area2D SwordHitbox;
00052     protected CollisionShape2D MainCollision;
00053     protected RayCast2D FrontCollisionRayCast;
00054     protected RayCast2D LineOfSight;
00055     protected RayCast2D LeftFallProtection;
00056     protected RayCast2D RightFallProtection;
00057     protected TextureProgressBar HealthBar;
00058     protected Player Player;
00059
00064     public override void _Ready()
00065     {
00066         Sprite = GetNode<AnimatedSprite2D>("AnimatedSprite2D");
00067         CollisionPolygon = GetNode<CollisionPolygon2D>("detection/CollisionPolygon2D");
00068         SwordHitbox = GetNode<Area2D>("AnimatedSprite2D/SwordHitBox");
00069         MainCollision = GetNode<CollisionShape2D>("MainCollision");
00070         FrontCollisionRayCast = GetNode<RayCast2D>("FrontCollisionRayCast");
00071         LineOfSight = GetNode<RayCast2D>("LineOfSight");
00072         LeftFallProtection = GetNode<RayCast2D>("LeftFallProtection");
00073         RightFallProtection = GetNode<RayCast2D>("RightFallProtection");
00074         HealthBar = GetNode<TextureProgressBar>("HealthBar");
00075         Player = GetNode<Player>("../../Player");
00076
00077         CurrentHealthPoints = MaxHealthPoints;
00078         CurrentStamina = MaxStamina;
00079         ReturnToStart = ReturnToStartAfter;
00080         StartPosition = Position;
00081         StartRotation = Sprite.FlipH;
00082
00083         HealthBar.Value = 100f* CurrentHealthPoints/MaxHealthPoints;
00084     }
00085
00091     public override void _Process(double DeltaTime)
00092     {
00093         HandleMovement(DeltaTime);
00094         if(CurrentStamina < MaxStamina){
00095             CurrentStamina += (float) DeltaTime;
00096             Velocity = Velocity * 0.8f;
00097         }
00098         if (!IsOnFloor() && !Dead) {
00099             Velocity += GetGravity() * (float)DeltaTime;
00100         }
00101         UpdateAnimation();
00102         MoveAndSlide();
00103         CheckPlayerHit();
00104     }
00105
00110     public void OnDetectionBodyEntered(Node2D body){
00111         if(CheckLineOfSight(body)){
00112             Pursuing = true;
00113             CurrentTarget = body;
00114         }
00115     }
00116
00121     public void OnPursuingRadiusBodyExited(Node2D body){
00122         if(body == CurrentTarget){
00123             Pursuing = false;
00124             CurrentTarget = null;
00125         }
00126     }
00127
00132     public void OnHitboxAreaEntered(Area2D area){
00133         Player Player1 = (Player) area.GetParent().GetParent();
00134         TakeDamage(Player1.GetDamage());
00135     }
00136
00141     public void OnSwordHitBoxBodyEntered(Node2D body){
00142         if(Dead) return;
00143         Sprite.Play("attack");
```

```
00144     }
00145
00150     private void HandleMovement(double DeltaTime){
00151         if(Dead) return;
00152         if((Sprite.Animation == "take_hit" || Sprite.Animation == "attack") && Sprite.IsPlaying()){
00153             Velocity = Vector2.Zero;
00154             return;
00155         }
00156         if(Pursuing){
00157             AnimationState = State.WALK;
00158             TargetPosition = CurrentTarget.Position;
00159             if(IsCloseTo(Position.X, TargetPosition.X, 0.1f)){
00160                 AnimationState = State.IDLE;
00161                 Velocity = Vector2.Zero;
00162                 return;
00163             }
00164             ReturnToStart = ReturnToStartAfter;
00165         } else if(ReturnToStart >= 0){
00166             AnimationState = State.IDLE;
00167             ReturnToStart -= DeltaTime;
00168             TargetPosition = Vector2.Inf;
00169         } else if(!IsCloseTo(Position.X, StartPosition.X, 0.1f)){
00170             AnimationState = State.WALK;
00171             TargetPosition = StartPosition;
00172         }
00173
00174         if(TargetPosition != Vector2.Inf){
00175
00176             if(IsCloseTo(Position.X, TargetPosition.X, 0.1f)){
00177                 AnimationState = State.IDLE;
00178                 Velocity = Vector2.Zero;
00179                 if(TargetPosition == StartPosition && Sprite.FlipH != StartRotation){
00180                     FlipRotation();
00181                 }
00182                 TargetPosition = Vector2.Inf;
00183                 return;
00184             }
00185
00186             if(TargetPosition.X > Position.X){
00187                 SetRotation(true);
00188                 if(!FrontCollisionRayCast.IsColliding()){
00189                     Vector2 velocity = Vector2.Zero;
00190                     velocity.X = Speed;
00191                     Velocity = velocity;
00192                 }
00193             } else {
00194                 SetRotation(false);
00195                 if(!FrontCollisionRayCast.IsColliding()){
00196                     Vector2 velocity = Vector2.Zero;
00197                     velocity.X = -Speed;
00198                     Velocity = velocity;
00199                 }
00200             }
00201
00202             if((!RightFallProtection.IsColliding() && !Sprite.FlipH) ||
      (!LeftFallProtection.IsColliding() && Sprite.FlipH)){
00203                 Velocity = Vector2.Zero;
00204             }
00205
00206         } else {
00207             Velocity = Vector2.Zero;
00208             AnimationState = State.IDLE;
00209         }
00210     }
00211
00212
00216     protected virtual void UpdateAnimation(){
00217         if(Dead) return;
00218         if(!((Sprite.Animation == "take_hit" || Sprite.Animation == "attack") && Sprite.IsPlaying())){
00219             switch(AnimationState){
00220                 case State.IDLE:
00221                     Sprite.Play("idle");
00222                     break;
00223
00224                 case State.WALK:
00225                     Sprite.Play("walk");
00226                     break;
00227
00228                 case State.ATTACK:
00229                     Sprite.Play("attack");
00230                     break;
00231
00232                 case State.TAKE_HIT:
00233                     Sprite.Play("take_hit");
00234                     break;
00235             }
00236         }
```

```
00237         HealthBar.Value = 100f* CurrentHealthPoints/MaxHealthPoints;
00238
00239     }
00240
00245     private void TakeDamage(Damage DMG){
00246         if(Dead) {
00247             return;
00248         }
00249         CurrentHealthPoints -= DMG.GetPhysicalDMG() * (1 - Armor / 100.0f) + DMG.GetTrueDMG();
00250         Position += DMG.GetPushAmount();
00251         if(CurrentHealthPoints <= 0){
00252             Die();
00253         } else {
00254             Sprite.Play("take_hit");
00255             if(DMG.GetSource() == Player){
00256                 Pursuing = true;
00257                 CurrentTarget = Player;
00258             }
00259         }
00260     }
00261
00266     public bool IsDead(){
00267         return Dead;
00268     }
00269
00274     private void CheckPlayerHit(){
00275         if(Dead) return;
00276         if(Sprite.Animation != "attack"){
00277             AlreadyHit = false;
00278             if(Sprite.Animation == "take_hit" || CurrentStamina < MaxStamina) return;
00279             Godot.Collections.Array<Node2D> Bodies = SwordHitbox.GetOverlappingBodies();
00280             foreach(Node2D Body in Bodies){
00281                 if(Body == Player){
00282                     Sprite.Play("attack");
00283                 }
00284             }
00285             return;
00286         }
00287         if(AlreadyHit) return;
00288         if(Sprite.Frame >= 6){
00289             CurrentStamina = 0;
00290             Godot.Collections.Array<Node2D> Bodies = SwordHitbox.GetOverlappingBodies();
00291             foreach(Node2D Body in Bodies){
00292                 if(Body == Player){
00293                     Player.TakeDamage(new Damage(Damage, 0f, Vector2.Zero, this));
00294                     AlreadyHit = true;
00295                     break;
00296                 }
00297             }
00298         }
00299
00300     }
00301
00305     private void Die(){
00306         Dead = true;
00307         Velocity = Vector2.Zero;
00308         MainCollision.SetDeferred(CollisionShape2D.PropertyName.Disabled, true);
00309
00310         Sprite.Play("death");
00311         HealthBar.SetVisible(false);
00312         Player.SetSinAmount(PlayerStats.Instance.GetSinAmount() + SinAmount);
00313
00314     }
00315
00319     public void Respawn()
00320     {
00321         Dead = false;
00322         CurrentHealthPoints = MaxHealthPoints;
00323         HealthBar.Value = 100f * CurrentHealthPoints / MaxHealthPoints;
00324         MainCollision.SetDeferred(CollisionShape2D.PropertyName.Disabled, false);
00325         HealthBar.SetVisible(true);
00326         Sprite.Play("idle");
00327     }
00328
00334     private bool CheckLineOfSight(Node2D body){
00335         Vector2 offset = Vector2.Zero;
00336         offset.Y = -14;
00337         LineOfSight.TargetPosition = body.Position + offset - (Position + LineOfSight.Position);
00338         if(LineOfSight.IsColliding()){
00339             return LineOfSight.GetCollider() == body;
00340         }
00341         return true;
00342     }
00343
00347     private void FlipRotation(){
00348         Sprite.FlipH = !Sprite.FlipH;
00349         CollisionPolygon.RotationDegrees = Math.Abs(CollisionPolygon.RotationDegrees -180);
```

```
00350         SwordHitbox.RotationDegrees = Math.Abs(SwordHitbox.RotationDegrees -180);
00351         FrontCollisionRayCast.RotationDegrees = Math.Abs(FrontCollisionRayCast.RotationDegrees - 180);
00352     }
00353
00358     private void SetRotation(bool Rotation){
00359         Sprite.FlipH = Rotation ^ StartRotation; // XOR mit StartRotation
00360         if(Rotation){
00361             CollisionPolygon.RotationDegrees = 180;
00362             SwordHitbox.RotationDegrees = 180;
00363             FrontCollisionRayCast.RotationDegrees = 180;
00364         } else {
00365             CollisionPolygon.RotationDegrees = 0;
00366             SwordHitbox.RotationDegrees = 0;
00367             FrontCollisionRayCast.RotationDegrees = 0;
00368         }
00369     }
00370
00378     private bool IsCloseTo(float Value1, float Value2, float Delta){
00379         return Value1 <= (Value2 + Delta) && Value1 >= (Value2 - Delta);
00380     }
00381 }
```

9.19 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/scripts/BloodVial.cs File Reference

Classes

- class BloodVial

  Klasse für die Interaktion zum heilen.

9.20 BloodVial.cs

Go to the documentation of this file.

```
00001 using GdMUT;
00002 using Godot;
00003 using System;
00004
00008 public partial class BloodVial : Label {
00009
00014     public override void _Ready() {
00015         Text = PlayerStats.Instance.GetBVCurrentUses() + "";
00016     }
00017
00021     public void UseBloodVial(){
00022         if(PlayerStats.Instance.GetBVCurrentUses() <= 0) return;
00023         PlayerStats.Instance.SetBVCurrentUses(PlayerStats.Instance.GetBVCurrentUses() - 1);
00024         Text = PlayerStats.Instance.GetBVCurrentUses() + "";
00025         PlayerStats.Instance.SetCurrentHealth(PlayerStats.Instance.GetCurrentHealth() +
      PlayerStats.Instance.GetBVHealAmount());
00026     }
00027
00031     public void ResetUses(){
00032         PlayerStats.Instance.SetBVCurrentUses(PlayerStats.Instance.GetBVMaxUses());
00033         Text = PlayerStats.Instance.GetBVCurrentUses() + "";
00034     }
00035
00040     public void AddMaxUses(int Amount){
00041         PlayerStats.Instance.SetBVMaxUses(PlayerStats.Instance.GetBVMaxUses() + Amount);
00042         ResetUses();
00043     }
00044
00048     public void LevelHealAmount(){
00049         PlayerStats.Instance.SetBVHealAmount(PlayerStats.Instance.GetBVHealAmount() + 25);
00050     }
00051 }
```

9.21 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/scripts/Boss1.cs File Reference

Classes

- class Boss1

  Klasse für einen stärkeren Boss-Gegner, der von BaseEnemy erbt.

9.22 Boss1.cs

Go to the documentation of this file.

```
00001 using Godot;
00002 using System;
00003
00007 public partial class Boss1 : BaseEnemy{
00008
00009     private bool EnemiesRevived = false;
00010     private float RegenCooldown = 5.0f; // Zeit, nach der Regeneration beginnt, wenn kein Schaden
    genommen wurde
00011     private float RegenTimer = 0.0f; // Timer für die Zeit seit dem letzten Angriff
00012     private float RegenAmount = 10.0f; // Menge an Gesundheit, die pro Tick regeneriert wird
00013
00014
00018     public override void _Ready(){
00019
00020         MaxHealthPoints = 400f;
00021         Damage = 50f;
00022         Armor = 30f;
00023         Speed = 10f;
00024         SinAmount = 100; // Bonuspunkte für Spieler beim Besiegen des Bosses
00025
00026         base._Ready();
00027
00028         CurrentHealthPoints = MaxHealthPoints;
00029         HealthBar.Value = 100f * CurrentHealthPoints / MaxHealthPoints;
00030     }
00031
00036     public override void _Process(double DeltaTime){
00037         base._Process(DeltaTime);
00038
00039         if (CurrentHealthPoints <= MaxHealthPoints / 2 && !EnemiesRevived){
00040             StartGlowing();
00041             ReviveEnemies();
00042             EnemiesRevived = true;
00043             Armor = 60f; // Rüstung erhöhen
00044         }
00045
00046         HandleRegeneration(DeltaTime);
00047     }
00048
00053     private void HandleRegeneration(double DeltaTime){
00054         if (CurrentHealthPoints < MaxHealthPoints){
00055             RegenTimer += (float)DeltaTime;
00056
00057             if (RegenTimer >= RegenCooldown){
00058                 CurrentHealthPoints = Math.Min(CurrentHealthPoints + RegenAmount, MaxHealthPoints);
00059                 HealthBar.Value = 100f * CurrentHealthPoints / MaxHealthPoints;
00060                 RegenTimer = 0.0f; // Timer zurücksetzen
00061             }
00062         }
00063     }
00064
00068     private void StartGlowing(){
00069         // Ändere die Modulationsfarbe des Sprites, um ein Leuchten zu simulieren
00070         if (Sprite != null){
00071             ShowPopupMessage("AHHHH!!!");
00072             Sprite.Modulate = new Color(1.0f, 0.84f, 0.0f, 1.0f); // Ein goldliche Leuchteffekt
00073         }
00074     }
00075
00080     private void ShowPopupMessage(string Message){
00081         Label popup = new Label();
00082         popup.Text = Message;
00083         popup.AddThemeColorOverride("font_color", new Color(1, 0, 0)); // Rot
00084         popup.Modulate = new Color(1, 1, 1, 0); // Start transparent
00085         popup.AutowrapMode = TextServer.AutowrapMode.Word;
```

```
00086         popup.SizeFlagsHorizontal = (Control.SizeFlags)(int)Control.SizeFlags.ExpandFill;
00087         popup.SizeFlagsVertical = (Control.SizeFlags)(int)Control.SizeFlags.ShrinkCenter;
00088         popup.HorizontalAlignment = HorizontalAlignment.Center;
00089         popup.VerticalAlignment = VerticalAlignment.Center;
00090 
00091 
00092         Vector2 bossGlobalPosition = GetGlobalTransformWithCanvas().Origin;
00093         popup.GlobalPosition = bossGlobalPosition + new Vector2(0, -100);
00094 
00095         CanvasLayer canvas = new CanvasLayer();
00096         AddChild(canvas);
00097         canvas.AddChild(popup);
00098 
00099         var tween = CreateTween();
00100         tween.TweenProperty(popup, "modulate:a", 1, 0.5f).From(0); // Einblenden
00101         tween.TweenProperty(popup, "modulate:a", 0, 0.5f).From(1).SetDelay(1.0f); // Ausblenden nach 1
      Sekunde
00102         tween.Connect("finished", new Callable(popup, "queue_free"));
00103     }
00104 
00108     private void ReviveEnemies()
00109     {
00110         // Hole den Elternknoten (bossRoom)
00111         Node BossRoom = GetParent();
00112 
00113         // Iteriere durch alle Kinder von bossRoom
00114         foreach (Node Child in BossRoom.GetChildren()){
00115             if (Child is BaseEnemy BaseEnemy && BaseEnemy.IsDead()){
00116                 BaseEnemy.Respawn();
00117             }
00118         }
00119     }
00120 }
```

9.23 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/scripts/Checkpoint.cs File Reference

Classes

- class Checkpoint

9.24 Checkpoint.cs

Go to the documentation of this file.

```
00001 using Godot;
00002 using System;
00003 
00004 public partial class Checkpoint : Node2D
00005 {
00006 
00007     // Variable für Player
00008     private Player Player;
00009 
00010     /*
00011      * @brief Intitalisierung der Node Player
00012      */
00013     public override void _Ready()
00014     {
00015         // Zugriff auf Player Node
00016         Player = GetNode<Player>("../Player");
00017     }
00018 
00019     /*
00020      * @brief Diese Funktion wird aufgerufen, wenn der Player den Checkpoint betritt
00021      * @param body Der Körper, der den Checkpoint betritt
00022      */
00023     private void OnPlayerBodyEntered(Node body)
00024     {
00025 
00031         if (body is Player Player)
00032         {
00033             // Setzen des Spawnpoints
00034             PlayerStats PlayerStats = GetNode<PlayerStats>("/root/PlayerStats");
```

```
00035             PlayerStats.Instance.SetSpawnPoint(this.GlobalPosition);
00036             Player.MaxHeal();
00037             PlayerStats.Instance.SetStamina(PlayerStats.Instance.GetMaxStamina());
00038             Player.GetBloodVials().ResetUses();
00039             GD.Print("Spawnpoint des Players gesetzt auf: ", this.GlobalPosition);
00040
00041             PlayerStats.SetRespawnLevelTag(GetParent().Name);
00042             GD.Print("RespawnLevelTag des Players gesetzt auf: ", GetParent().Name);
00043             GD.Print(PlayerStats.Instance.GetRespawnLevelTag());
00044         }
00045
00046     }
00047 }
```

9.25 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/scripts/Damage.cs File Reference

Classes

- class Damage

  Repräsentiert den Schaden, der von Charakteren oder Gegnern verursacht wird. Beinhaltet physischen Schaden, wahren Schaden und den Rückstoßeffekt.

9.26 Damage.cs

Go to the documentation of this file.

```
00001 using Godot;
00002
00007 public class Damage{
00008
00009     private float PhysicalDMG;
00010     private float TrueDMG;
00011     private Vector2 PushAmount;
00012     private Node2D Source;
00013
00020     public Damage(float PhysicalDMG, float TrueDMG, Vector2 PushAmount, Node2D Source){
00021         this.PhysicalDMG = PhysicalDMG;
00022         this.TrueDMG = TrueDMG;
00023         this.PushAmount = PushAmount;
00024         this.Source = Source;
00025     }
00026
00031     public float GetPhysicalDMG(){
00032         return PhysicalDMG;
00033     }
00034
00039     public float GetTrueDMG(){
00040         return TrueDMG;
00041     }
00042
00047     public Vector2 GetPushAmount(){
00048         return PushAmount;
00049     }
00050
00055     public Node2D GetSource(){
00056         return Source;
00057     }
00058 }
```

9.27 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/scripts/Door.cs File Reference

Classes

- class Door

  Klasse für die Tür.

9.28 Door.cs

Go to the documentation of this file.

```
00001 using Godot;
00002 using System;
00003
00008 public partial class Door : Area2D
00009 {
00010     public Node Spawn;
00011
00012     [Export]
00013     public string DestinationLevelTag { get; set; }
00014
00015     [Export]
00016     public string DestinationDoorTag { get; set; }
00017
00018     [Export]
00019     public string SpawnDirection { get; set; } = "up";
00020
00021
00022
00026     public override void _Ready()
00027     {
00028         Spawn = GetNode("Spawn");
00029     }
00030
00031
00036     private void OnPlayerBodyEntered(Node body)
00037     {
00038         if (body is Player player)
00039         {
00040             var NavigationManager = GetNode<NavigationManager>("/root/NavigationManager");
00041             NavigationManager.GoToLevel(DestinationLevelTag, DestinationDoorTag);
00042         }
00043     }
00044 }
```

9.29 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/scripts/HealthBar.cs File Reference

Classes

- class HealthBar

  Klasse für die Gesundheitsleiste des Spielers. Synchronisiert die Anzeige der HealthBar mit den Lebenspunkten des Spielers.

9.30 HealthBar.cs

Go to the documentation of this file.

```
00001 using Godot;
00002
00007 public partial class HealthBar : TextureProgressBar {
00008
00013     public override void _Ready() {
00014         // Setze die maximale Gesundheit der HealthBar basierend auf der Spieler-MaxHealth
00015         MaxValue = PlayerStats.Instance.GetMaxHealthPoints();
00016         Value = PlayerStats.Instance.GetCurrentHealth();
00017     }
00018
00024     public override void _Process(double DeltaTime) {
00025         // Aktualisiere den Wert der HealthBar basierend auf der aktuellen Gesundheit des Spielers
00026         Value = PlayerStats.Instance.GetCurrentHealth();
00027     }
00028 }
```

9.31 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/scripts/Hud.cs File Reference

Classes

- class Hud

  Klasse für das PauseMenu.

9.32 Hud.cs

Go to the documentation of this file.

```
00001 using Godot;
00002 using System;
00003
00004
00008 public partial class Hud : CanvasLayer {
00009
00010     private AnimationPlayer AnimationPlayer;
00011     private CenterContainer Buttons;
00012     private bool Enabled;
00013
00014
00019     public override void _Ready() {
00020         AnimationPlayer = GetNode<AnimationPlayer>("PauseMenu/AnimationPlayer");
00021         Buttons = GetNode<CenterContainer>("PauseMenu/Buttons");
00022         AnimationPlayer.Play("RESET");
00023     }
00024
00029     public override void _Process(double DeltaTime) {
00030         if(Input.IsActionJustPressed("escape")){
00031             TogglePause();
00032         }
00033     }
00034
00038     private void TogglePause(){
00039         Enabled = !Enabled;
00040         GetTree().Paused = Enabled;
00041         if(Enabled){
00042             AnimationPlayer.Play("Pause");
00043             Buttons.Visible = true;
00044         } else {
00045             AnimationPlayer.PlayBackwards("Pause");
00046             Buttons.Visible = false;
00047         }
00048     }
00049
00053     public void OnResumeButtonPressed(){
00054         TogglePause();
00055     }
00056
00060     public void OnSaveButtonPressed(){
00061         StorageManager.Instance.SaveAll(StorageManager.Instance.GetLastSaveId());
00062     }
00063
00067     public void OnSaveMenuButtonPressed(){
00068         StorageManager.Instance.SaveAll(StorageManager.Instance.GetLastSaveId());
00069         NavigationManager.Instance.GoToLevel("main_menu", null);
00070         PlayerStats.Instance.Reload();
00071         GetTree().Paused = false;
00072     }
00073
00077     public void OnSaveQuitButtonPressed(){
00078         StorageManager.Instance.SaveAll(StorageManager.Instance.GetLastSaveId());
00079         GetTree().Quit();
00080     }
00081
00082 }
```

9.33 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/scripts/Interactable.cs File Reference

Classes

- class Interactable

Klasse für Interaktion.

9.34 Interactable.cs

Go to the documentation of this file.

```
00001 using Godot;
00002 using System;
00003
00007 public partial class Interactable : AnimatedSprite2D {
00008
00009     private Player Player;
00010     private RichTextLabel TextLabel;
00011     private Control PopUp;
00012     private Area2D Area;
00013
00014     [Export(PropertyHint.MultilineText)]
00015     private String Text { get; set;}
00016
00021     public override void _Ready(){
00022         Player = GetNode<Player>("../Player");
00023         TextLabel = GetNode<RichTextLabel>("../HUD/PopUp/Text");
00024         PopUp = GetNode<Control>("../HUD/PopUp");
00025         Area = GetNode<Area2D>("Area2D");
00026     }
00027
00032     public override void _Process(double DeltaTime){
00033         if(Input.IsActionJustPressed("interact")){
00034             Godot.Collections.Array<Node2D> Bodies = Area.GetOverlappingBodies();
00035             foreach(Node2D Body in Bodies){
00036                 if(Body == Player){
00037                     TextLabel.Clear();
00038                     TextLabel.AppendText(Text);
00039                     PopUp.Visible = true;
00040                     return;
00041                 }
00042             }
00043         }
00044     }
00045
00050     public void OnAreaBodyExited(Node2D Body){
00051         if(Body == Player){
00052             PopUp.Visible = false;
00053             TextLabel.Clear();
00054         }
00055     }
00056
00057 }
```

9.35 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/scripts/LevelManager.cs File Reference

Classes

- class LevelManager

  Klasse für den LevelManager Diese Klasse verwaltet den Levelwechsel und die Spielerpositionierung.

9.36 LevelManager.cs

Go to the documentation of this file.

```
00001 using Godot;
00002
00007 public partial class LevelManager : Node2D
00008 {
00013     public override void _Ready()
00014     {
00015         var NavigationManager = GetNode<NavigationManager>("/root/NavigationManager");
```

```
00016
00021         if (NavigationManager.SpawnDoorTag != null)
00022         {
00023             OnLevelSpawn(NavigationManager.SpawnDoorTag);
00024         }
00025         else
00026         {
00027             NavigationManager.CallDeferred("TriggerPlayerSpawn", PlayerStats.Instance.GetPosition(),
    "");
00028         }
00029
00030     }
00031
00036     private void OnLevelSpawn(string DestinationTag)
00037     {
00038         var NavigationManager = GetNode<NavigationManager>("/root/NavigationManager");
00039         // Pfad zur Tür basierend auf dem Ziel-Tag erstellen
00040         string DoorPath = "Doors/Door_" + DestinationTag;
00041
00042         Door door = GetNode<Door>(DoorPath);
00043
00044         // TriggerPlayerSpawn nach deferred ausführen
00045         NavigationManager.CallDeferred("TriggerPlayerSpawn", door.GlobalPosition,
    door.SpawnDirection);
00046     }
00047 }
```

9.37 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/scripts/MainMenu.cs File Reference

Classes

- class MainMenu

  Klasse für das MainMenu.

9.38 MainMenu.cs

Go to the documentation of this file.

```
00001 using Godot;
00002 using System;
00003
00007 public partial class MainMenu : Node2D {
00008
00009     private int MenuState = 0;
00010     private VBoxContainer Navigation;
00011     private MarginContainer SavesContainer;
00012     private Button ContinueButton;
00013     private Label InfoLabel;
00014     private Label[] SaveLabel = new Label[3];
00015     private Button[] SelectButton = new Button[3];
00016     private Button[] DeleteButton = new Button[3];
00017     private ConfirmationDialog DeleteConfirmation;
00018     private int SaveToDelete = 0;
00019
00020
00025     public override void _Ready() {
00026         Navigation = GetNode<VBoxContainer>("Control/Navigation");
00027         SavesContainer = GetNode<MarginContainer>("Control/Saves");
00028         ContinueButton = GetNode<Button>("Control/Navigation/ContinueButton");
00029         InfoLabel = GetNode<Label>("Control/Saves/VBoxContainer/Info");
00030
00031         SaveLabel[0] = GetNode<Label>("Control/Saves/VBoxContainer/HBoxContainer/Save1/Label");
00032         SelectButton[0] = GetNode<Button>("Control/Saves/VBoxContainer/HBoxContainer/Save1/Select");
00033         DeleteButton[0] = GetNode<Button>("Control/Saves/VBoxContainer/HBoxContainer/Save1/Delete");
00034         SaveLabel[1] = GetNode<Label>("Control/Saves/VBoxContainer/HBoxContainer/Save2/Label");
00035         SelectButton[1] = GetNode<Button>("Control/Saves/VBoxContainer/HBoxContainer/Save2/Select");
00036         DeleteButton[1] = GetNode<Button>("Control/Saves/VBoxContainer/HBoxContainer/Save2/Delete");
00037         SaveLabel[2] = GetNode<Label>("Control/Saves/VBoxContainer/HBoxContainer/Save3/Label");
00038         SelectButton[2] = GetNode<Button>("Control/Saves/VBoxContainer/HBoxContainer/Save3/Select");
00039         DeleteButton[2] = GetNode<Button>("Control/Saves/VBoxContainer/HBoxContainer/Save3/Delete");
00040
00041         DeleteConfirmation = GetNode<ConfirmationDialog>("DeleteConfirmation");
```

```
00042
00043         if(StorageManager.Instance.GetLastSaveId() > -1){
00044             ContinueButton.Visible = true;
00045         }
00046     }
00047
00048
00052     private void Change(){
00053         if(MenuState == 0){
00054             SavesContainer.Visible = false;
00055             Navigation.Visible = true;
00056         } else {
00057             Navigation.Visible = false;
00058             SavesContainer.Visible = true;
00059
00060             int Saves = StorageManager.Instance.GetSaves();
00061
00062             if(MenuState == 1){
00063                 InfoLabel.Text = "Select empty save to start a new Game";
00064                 for(int i = 0; i < 3; i++){
00065                     if((Saves & (int) Math.Pow(2, i)) == (int) Math.Pow(2, i)){
00066                         SaveLabel[i].Text = "Save " + (i+1);
00067                         SelectButton[i].Disabled = true;
00068                         DeleteButton[i].Disabled = false;
00069                     } else {
00070                         SaveLabel[i].Text = "Save " + (i+1) + "\nEmpty";
00071                         SelectButton[i].Disabled = false;
00072                         DeleteButton[i].Disabled = true;
00073                     }
00074                 }
00075             } else {
00076                 InfoLabel.Text = "Select save to load Game";
00077                 for(int i = 0; i < 3; i++){
00078                     if((Saves & (int) Math.Pow(2, i)) == (int) Math.Pow(2, i)){
00079                         SaveLabel[i].Text = "Save " + (i+1);
00080                         SelectButton[i].Disabled = false;
00081                         DeleteButton[i].Disabled = false;
00082                     } else {
00083                         SaveLabel[i].Text = "Save " + (i+1) + "\nEmpty";
00084                         SelectButton[i].Disabled = true;
00085                         DeleteButton[i].Disabled = true;
00086                     }
00087                 }
00088             }
00089         }
00090     }
00091
00095     public void OnContinueButtonPressed(){
00096         StorageManager.Instance.LoadGameFile(StorageManager.Instance.GetLastSaveId());
00097         NavigationManager.Instance.GoToLevel(PlayerStats.Instance.GetCurrentLevelTag(), null);
00098     }
00099
00103     public void OnQuitButtonPressed(){
00104         StorageManager.Instance.SaveSettings();
00105         GetTree().Quit();
00106     }
00107
00111     public void OnNewGameButtonPressed(){
00112         MenuState = 1;
00113         Change();
00114     }
00115
00119     public void OnLoadGameButtonPressed(){
00120         MenuState = 2;
00121         Change();
00122     }
00123
00127     public void OnBackButtonPressed(){
00128         MenuState = 0;
00129         Change();
00130     }
00131
00135     public void OnSave1SelectPressed(){
00136         if(MenuState == 2){
00137             StorageManager.Instance.LoadGameFile(0);
00138         }
00139         NavigationManager.Instance.GoToLevel(PlayerStats.Instance.GetCurrentLevelTag(), null);
00140         StorageManager.Instance.SetSaves(StorageManager.Instance.GetSaves() | 1);
00141         StorageManager.Instance.SetLastSaveId(0);
00142     }
00143
00147     public void OnSave1DeletePressed(){
00148         SaveToDelete = 1;
00149         DeleteConfirmation.SetText("Are you sure you want to DELETE Save " + SaveToDelete + "?");
00150         DeleteConfirmation.Show();
00151     }
00152
```

```
00156     public void OnSave2SelectPressed(){
00157         if(MenuState == 2){
00158             StorageManager.Instance.LoadGameFile(1);
00159         }
00160         NavigationManager.Instance.GoToLevel(PlayerStats.Instance.GetCurrentLevelTag(), null);
00161         StorageManager.Instance.SetSaves(StorageManager.Instance.GetSaves() | 2);
00162         StorageManager.Instance.SetLastSaveId(1);
00163     }
00164 
00168     public void OnSave2DeletePressed(){
00169         SaveToDelete = 2;
00170         DeleteConfirmation.SetText("Are you sure you want to DELETE Save " + SaveToDelete + "?");
00171         DeleteConfirmation.Show();
00172     }
00173 
00177     public void OnSave3SelectPressed(){
00178         if(MenuState == 2){
00179             StorageManager.Instance.LoadGameFile(2);
00180         }
00181         NavigationManager.Instance.GoToLevel(PlayerStats.Instance.GetCurrentLevelTag(), null);
00182         StorageManager.Instance.SetSaves(StorageManager.Instance.GetSaves() | 4);
00183         StorageManager.Instance.SetLastSaveId(2);
00184     }
00185 
00189     public void OnSave3DeletePressed(){
00190         SaveToDelete = 3;
00191         DeleteConfirmation.SetText("Are you sure you want to DELETE Save " + SaveToDelete + "?");
00192         DeleteConfirmation.Show();
00193     }
00194 
00198     public void OnDeleteConfirmationCanceled(){
00199         SaveToDelete = 0;
00200         Change();
00201     }
00202 
00206     public void OnDeleteConfirmationConfirmed(){
00207         StorageManager.Instance.SetSaves(StorageManager.Instance.GetSaves() ^ (int) Math.Pow(2,
      SaveToDelete - 1));
00208         Change();
00209     }
00210 
00214     public void OnDeleteConfirmationCloseRequested(){
00215         OnDeleteConfirmationCanceled();
00216     }
00217 
00218 }
```

9.39 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/scripts/MainMenuBackground.cs File Reference

Classes

- class MainMenuBackground

  Klasse für die MainMenuBackground-Animation.

9.40 MainMenuBackground.cs

Go to the documentation of this file.

```
00001 using Godot;
00002 using System;
00003 
00007 public partial class MainMenuBackground : ParallaxLayer {
00008 
00009     [Export]
00010     private float ScrollSpeed = -10f;
00011 
00016     public override void _Process(double DeltaTime) {
00017         float X = GetMotionOffset().X;
00018         X += ScrollSpeed * (float) DeltaTime;
00019         SetMotionOffset(new Vector2(X,0));
00020     }
00021 }
```

9.41 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/scripts/NavigationManager.cs File Reference

Classes

- class NavigationManager

  Der NavigationManager ist für das Laden von Leveln und das Spawnen des Spielers verantwortlich. Der NavigationManager ist ein Singleton, der in der Haupt-Szene platziert wird und von anderen Skripten verwendet wird, um Level zu laden und den Spieler zu spawnen.

9.42 NavigationManager.cs

Go to the documentation of this file.

```
00001 using Godot;
00002
00007 public partial class NavigationManager : Node
00008 {
00009     public static NavigationManager Instance { get; private set; }
00010     // Deklarieren der vorab geladenen Szenen
00011     private static readonly PackedScene SceneMainMenu =
      (PackedScene)GD.Load("res://Scenes/main_menu.tscn");
00012     private static readonly PackedScene SceneIntro = (PackedScene)GD.Load("res://Scenes/intro.tscn");
00013     private static readonly PackedScene SceneLevel1 =
      (PackedScene)GD.Load("res://Scenes/level1.tscn");
00014     private static readonly PackedScene SceneBoss =
      (PackedScene)GD.Load("res://Scenes/bossRoom.tscn");
00015     private static readonly PackedScene SceneLevelOne =
      (PackedScene)GD.Load("res://Scenes/level_one.tscn");
00016     private static readonly PackedScene SceneLevelTwo =
      (PackedScene)GD.Load("res://Scenes/level_two.tscn");
00017
00018     // Die Spawn-Tag-Variable
00019     public string SpawnDoorTag { get; private set; }
00020
00026     [Signal]
00027     public delegate void OnTriggerPlayerSpawnEventHandler(Vector2 Position, string Direction);
00028
00032     public override void _Ready(){
00033         Instance = this;
00034     }
00035
00041     public void GoToLevel(string LevelTag, string DestinationTag)
00042     {
00043         PackedScene SceneToLoad = null;
00044
00045         // Bestimmen, welches Level geladen werden soll
00046         switch (LevelTag)
00047         {
00048             case "main_menu":
00049                 SceneToLoad = SceneMainMenu;
00050                 break;
00051             case "intro":
00052                 SceneToLoad = SceneIntro;
00053                 break;
00054             case "level1":
00055                 SceneToLoad = SceneLevel1;
00056                 break;
00057             case "bossRoom":
00058                 SceneToLoad = SceneBoss;
00059                 break;
00060             case "level_one":
00061                 SceneToLoad = SceneLevelOne;
00062                 break;
00063             case "level_two":
00064                 SceneToLoad = SceneLevelTwo;
00065                 break;
00066         }
00067
00068         // Überprüfen, ob eine Szene ausgewählt wurde und diese dann laden
00069         if (SceneToLoad != null){
00070             if(SceneToLoad != SceneMainMenu){
00071                 PlayerStats.Instance.SetCurrentLevelTag(LevelTag);
00072                 SpawnDoorTag = DestinationTag;
```

```
00073             }
00074             // Verwendung der ChangeSceneToPacked-Methode in Godot 4
00075             CallDeferred(nameof(DeferredChangeScene), SceneToLoad);
00076         }
00077     }
00078
00083     private void DeferredChangeScene(PackedScene SceneToLoad)
00084     {
00085         GetTree().ChangeSceneToPacked(SceneToLoad);
00086     }
00087
00093     public void TriggerPlayerSpawn(Vector2 Position, string Direction)
00094     {
00095         EmitSignal(SignalName.OnTriggerPlayerSpawn, Position, Direction);
00096     }
00097 }
```

9.43 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Player.cs File Reference

Classes

- class Player

  Klasse für den Spielercharakter. Verwaltet Bewegung, Sprünge, Angriffe und Animationen.

9.44 Player.cs

Go to the documentation of this file.

```
00001 using Godot;
00002 using System;
00003
00008 public partial class Player : CharacterBody2D
00009 {
00010     // Variablen für Bewegung, Sprünge und Dash
00011     private const float SPEED = 100f;
00012     private const float JUMP_VELOCITY = -300f;
00013     private int JumpMax = 2;
00014     private int JumpCount = 0;
00015
00016     private Vector2 DashDirection = Vector2.Zero;
00017     private float DashSpeed = 300f;
00018     private bool IsDashing = false;
00019     private bool CanDash = true;
00020     private float DashTrailInterval = 0.05f;
00021     private float DashTrailTimer = 0f;
00022
00023     // Referenzen zu den Knoten
00024     private AnimationPlayer AnimationPlayer;
00025     private Sprite2D Sprite;
00026     private Timer DashEffect;
00027     private Timer DashTimer;
00028     private CollisionShape2D SwordCollision;
00029     private CollisionShape2D PlayerHitbox;
00030     private BloodVial BloodVials;
00031     private Label SinDisplay;
00032
00033     private Vector2 HauptHitbox;
00034     private int LastAttack = 0;
00035
00036     //Variablen für Stamina
00037     private float TimeSinceLastStaminaUse = 0f;
00038
00043     public override void _Ready() {
00044         AnimationPlayer = GetNode<AnimationPlayer>("AnimationPlayer");
00045         Sprite = GetNode<Sprite2D>("Sprite2D");
00046         DashEffect = GetNode<Timer>("DashEffect");
00047         DashTimer = GetNode<Timer>("DashTimer");
00048         SwordCollision = GetNode<CollisionShape2D>("Sprite2D/SwordHit/SwordCollision");
00049         PlayerHitbox = GetNode<CollisionShape2D>("PlayerHitbox");
00050         HauptHitbox = PlayerHitbox.Position;
00051         BloodVials = GetNode<BloodVial>("../HUD/BloodVial/Counter");
00052         SinDisplay = GetNode<Label>("../HUD/SinAmount/Counter");
```

```
00053
00054         SinDisplay.Text = PlayerStats.Instance.GetSinAmount() + "";
00055
00056         NavigationManager navigationManager = GetNode<NavigationManager>("/root/NavigationManager");
00057         navigationManager.Connect("OnTriggerPlayerSpawn", new Callable(this, nameof(OnSpawn)));
00058
00059         Position = PlayerStats.Instance.GetPosition();
00060     }
00061
00067     public override void _PhysicsProcess(double DeltaTime) {
00068         // Gravitation hinzufügen, wenn der Charakter nicht am Boden ist
00069         if (!IsOnFloor()) {
00070             Velocity += GetGravity() * (float)DeltaTime;
00071         } else {
00072             CanDash = true; // Dash wird zurückgesetzt, wenn der Charakter am Boden ist
00073         }
00074
00075         TimeSinceLastStaminaUse += (float)DeltaTime;
00076         RegenerateStamina(20f, DeltaTime);
00077
00078         // Heal
00079         if(Input.IsActionJustPressed("heal")){
00080             BloodVials.UseBloodVial();
00081         }
00082
00083         HandleJump();
00084         HandleMovement(DeltaTime);
00085         MoveAndSlide();
00086         UpdateAnimations();
00087         PlayerStats.Instance.SetPosition(Position);
00088         }
00089
00094     private void HandleJump() {
00095         // Sprungzähler zurücksetzen, wenn der Charakter am Boden ist
00096         if (JumpCount != 0 && IsOnFloor()) {
00097             JumpCount = 0;
00098         }
00099
00100         // Überprüfen, ob der Sprung-Button gedrückt wurde und der Charakter noch Sprünge übrig hat
00101         if (Input.IsActionJustPressed("ui_up") && JumpCount < JumpMax) {
00102             if (JumpCount == 0) {
00103             // Erster Sprung ohne Stamina-Verlust
00104             Velocity = new Vector2(Velocity.X, JUMP_VELOCITY);
00105             JumpCount += 1;
00106             } else if (JumpCount > 0) {
00107                 // Beim Doppelsprung Stamina prüfen und abziehen
00108                 if (UseStamina(15)) {
00109                     Velocity = new Vector2(Velocity.X, JUMP_VELOCITY);
00110                     JumpCount += 1;
00111                 }
00112             }
00113         }
00114     }
00115
00121     private void HandleMovement(double DeltaTime) {
00122         Vector2 direction = new Vector2(Input.GetAxis("ui_left", "ui_right"), Input.GetAxis("ui_up",
      "ui_down")).Normalized();
00123         float currentSpeed = SPEED;
00124
00125         // Sprite umdrehen basierend auf der Bewegungsrichtung und Kollision umdrehen
00126         if (direction.X < 0) {
00127             Sprite.FlipH = true;
00128             SwordCollision.Position = new Vector2(-Mathf.Abs(SwordCollision.Position.X),
      SwordCollision.Position.Y);
00129             PlayerHitbox.Position = new Vector2(Sprite.Position.X * 1.8f, PlayerHitbox.Position.Y);
00130         } else if (direction.X > 0) {
00131             Sprite.FlipH = false;
00132             SwordCollision.Position = new Vector2(Mathf.Abs(SwordCollision.Position.X),
      SwordCollision.Position.Y);
00133             PlayerHitbox.Position = HauptHitbox;
00134         }
00135
00136         // Geschwindigkeit reduzieren, wenn der Spieler angreift
00137         if (AnimationPlayer.CurrentAnimation == "light_attack") {
00138             currentSpeed *= 0.5f;
00139         } else if (AnimationPlayer.CurrentAnimation == "heavy_attack") {
00140             currentSpeed *= 0.15f;
00141         }
00142
00143         // Blockieren stoppt die Bewegung
00144         if (IsBlocking()) {
00145             currentSpeed = 0;
00146         }
00147
00148         if (IsDashing) {
00149             DashInProgress(DeltaTime);
00150         } else {
```

```
00151             // Normale Bewegung verarbeiten, wenn kein Dash aktiv ist
00152             if (direction != Vector2.Zero) {
00153                 Velocity = new Vector2(direction.X * currentSpeed, Velocity.Y);
00154             } else {
00155                 Velocity = new Vector2(Mathf.MoveToward(Velocity.X, 0, SPEED), Velocity.Y);
00156             }
00157 
00158             // Überprüfen, ob der Dash-Button gedrückt wurde mit eine Bewegungsrichtung und nicht
      schon am angreifen ist
00159             if (Input.IsActionJustPressed("dash") && direction != Vector2.Zero && CanDash &&
      !IsAttacking()) {
00160                 // Wenn der Player genug Stamina hat kann er dashen
00161                 if (UseStamina(20)){
00162                     DashDirection = direction;
00163                     StartDash();
00164                 }
00165             }
00166         }
00167     }
00168 
00172     private void StartDash() {
00173         SetCollisionLayerValue(1,false);
00174         SetCollisionMaskValue(1,false);
00175         IsDashing = true;
00176         CanDash = false;
00177         DashTimer.Timeout += StopDash;
00178         DashTimer.Start();
00179         DashEffect.Start();
00180         DashTrailTimer = 0f;
00181     }
00182 
00187     private void DashInProgress(double DeltaTime) {
00188         // Charakter bewegt sich in die Dash-Richtung mit Dash-Geschwindigkeit
00189         if (DashDirection == Vector2.Up) {
00190             Velocity = DashDirection / 1.5f * DashSpeed;
00191         } else {
00192             Velocity = DashDirection * DashSpeed;
00193         }
00194 
00195         // Dash-Trail bei Intervallen erstellen
00196         DashTrailTimer -= (float)DeltaTime;
00197         if (DashTrailTimer <= 0f) {
00198             CreateDashEffect();
00199             DashTrailTimer = DashTrailInterval;
00200         }
00201     }
00202 
00207     private void CreateDashEffect() {
00208         Sprite2D PlayerCopyNode = (Sprite2D)Sprite.Duplicate();
00209         GetParent().AddChild(PlayerCopyNode);
00210 
00211         CollisionShape2D SwordCollisionCopy =
      PlayerCopyNode.GetNode<CollisionShape2D>("SwordHit/SwordCollision");
00212         if (SwordCollisionCopy != null) {
00213             SwordCollisionCopy.Disabled = true; // Deaktiviere die Kollision der Kopie
00214         }
00215 
00216         PlayerCopyNode.GlobalPosition = GlobalPosition + new Vector2(0, Sprite.Texture.GetHeight() *
      Sprite.Scale.Y * -0.5f);
00217 
00218         // Verblassen-Effekt für den Dash-Trail hinzufügen
00219         float AnimationTime = (float)(DashTimer.WaitTime / 3);
00220 
00221         Timer FadeTimer1 = new Timer();
00222         AddChild(FadeTimer1);
00223         FadeTimer1.Timeout += () => {
00224             if (IsInstanceValid(PlayerCopyNode)) {
00225                 PlayerCopyNode.Modulate = new Color(PlayerCopyNode.Modulate, 0.4f);
00226             }
00227         };
00228         FadeTimer1.Start(AnimationTime);
00229 
00230         Timer FadeTimer2 = new Timer();
00231         AddChild(FadeTimer2);
00232         FadeTimer2.Timeout += () => {
00233             if (IsInstanceValid(PlayerCopyNode)) {
00234                 PlayerCopyNode.Modulate = new Color(PlayerCopyNode.Modulate, 0.2f);
00235             }
00236         };
00237         FadeTimer2.Start(AnimationTime * 2);
00238 
00239         Timer FadeTimer3 = new Timer();
00240         AddChild(FadeTimer3);
00241         FadeTimer3.Timeout += () => {
00242             if (IsInstanceValid(PlayerCopyNode)) {
00243                 PlayerCopyNode.QueueFree();
00244             }
```

```
00245         };
00246         FadeTimer3.Start(AnimationTime * 3);
00247     }
00248
00252     private void StopDash() {
00253         IsDashing = false;
00254         DashEffect.Stop();
00255         DashTimer.Stop();
00256         DashTimer.Timeout -= StopDash;
00257         SetCollisionLayerValue(1,true);
00258         SetCollisionMaskValue(1,true);
00259     }
00260
00265     private bool IsAttacking() {
00266         return AnimationPlayer.CurrentAnimation == "heavy_attack" || AnimationPlayer.CurrentAnimation
      == "light_attack";
00267     }
00268
00273     private bool IsBlocking() {
00274         return AnimationPlayer.CurrentAnimation == "block";
00275     }
00276
00280     public void MaxHeal(){
00281         PlayerStats.Instance.SetCurrentHealth(PlayerStats.Instance.GetMaxHealthPoints());
00282     }
00283
00289     public void TakeDamage(Damage Damage){
00290         float totalDamage = Damage.GetTrueDMG();
00291         if(!IsBlocking()){
00292             totalDamage += Damage.GetPhysicalDMG();
00293         } else {
00294             float CurrentStamina = PlayerStats.Instance.GetStamina();
00295             CurrentStamina -= Damage.GetPhysicalDMG();
00296             if(CurrentStamina < 0){
00297                 totalDamage -= CurrentStamina;
00298             }
00299             PlayerStats.Instance.SetStamina(CurrentStamina);
00300         }
00301
00302         PlayerStats.Instance.SetCurrentHealth(PlayerStats.Instance.GetCurrentHealth() - totalDamage);
00303         Position += Damage.GetPushAmount();
00304
00305         // Überprüfe, ob der Spieler gestorben ist
00306         if (PlayerStats.Instance.GetCurrentHealth() <= 0){
00307             GD.Print("Spieler ist gestorben!");
00308             Respawn();
00309         }
00310     }
00311
00317     public Damage GetDamage(){
00318         if(LastAttack == 1){
00319             return new Damage(50, 0, Vector2.Zero, this);
00320         }
00321         if(LastAttack == 2){
00322             Vector2 Push = new Vector2(20,0);
00323             if(Sprite.FlipH){
00324                 Push = -Push;
00325             }
00326             return new Damage(100, 0, Push, this);
00327         }
00328         return new Damage(0,0,Vector2.Zero, this);
00329     }
00330
00336     public void RegenerateStamina(float Amount, double delta) {
00337         // Wenn die Verzögerungszeit erreicht wurde, regeneriere Stamina
00338         if (TimeSinceLastStaminaUse >= 1f) {
00339             PlayerStats.Instance.SetStamina(PlayerStats.Instance.GetStamina() + Amount *
      (float)delta); // Regeneriere Stamina abhängig von der Zeit
00340         }
00341     }
00342
00349     public bool UseStamina(float Amount) {
00350         // Versucht, eine bestimmte Menge an Stamina zu verbrauchen.
00351         // Gibt true zurück, wenn genug Stamina verfügbar war; andernfalls false.
00352         if (PlayerStats.Instance.GetStamina() >= Amount) {
00353             PlayerStats.Instance.SetStamina(PlayerStats.Instance.GetStamina() - Amount);
00354             TimeSinceLastStaminaUse = 0f;
00355             return true;
00356         }
00357
00358         return false;
00359     }
00360
00365     public void SlowPlayer(float SlowAmount){
00366         Velocity = new Vector2(Velocity.X * SlowAmount, Velocity.Y);
00367     }
00368
```

```
00372     public void Respawn(){
00373         var NavigationManager = GetNode<NavigationManager>("/root/NavigationManager");
00374         NavigationManager.GoToLevel(PlayerStats.Instance.GetRespawnLevelTag(), "spawn");
00375         BloodVials.ResetUses();
00376 
00377     }
00378 
00383     public BloodVial GetBloodVials(){
00384         return BloodVials;
00385     }
00386 
00391     public void SetSinAmount(int Value) {
00392         // SinAmount muss immer >= 0 sein
00393         PlayerStats.Instance.SetSinAmount(Value);
00394         SinDisplay.Text = PlayerStats.Instance.GetSinAmount() + "";
00395     }
00396 
00402     private void OnSpawn(Vector2 position, string direction){
00403 
00404         // Spielerposition auf die übergebene Position setzen
00405         if (direction == "right")
00406         {
00407             // Update the x value by adding 50 to it, keep the original y value
00408             Sprite.FlipH = false;
00409             position = position with { X = position.X + 25 };
00410         }
00411         else if (direction == "left")
00412         {
00413             // Update the x value by subtracting 50 from it, keep the original y value
00414             Sprite.FlipH = true;
00415             position = position with { X = position.X - 25 };
00416         }
00417         Position = position;
00418 
00419     }
00420 
00421 
00425     private void UpdateAnimations() {
00426         if (Input.IsActionJustPressed("light_attack") && !IsDashing && !IsAttacking()) {
00427             if (UseStamina(10)){
00428                 LastAttack = 1;
00429                 AnimationPlayer.Play("light_attack");
00430             }
00431         } else if (Input.IsActionJustPressed("heavy_attack") && !IsDashing && !IsAttacking()) {
00432             if (UseStamina(25)){
00433                 LastAttack = 2;
00434                 AnimationPlayer.Play("heavy_attack");
00435             }
00436         }
00437         if (Input.IsActionPressed("block") && !IsDashing && !IsAttacking() && IsOnFloor()) {
00438             if (UseStamina(0)){
00439                 AnimationPlayer.Play("block");
00440                 LastAttack = 0;
00441             }
00442         }
00443 
00444         if (IsOnFloor() && !IsAttacking() && !IsBlocking()) {
00445             LastAttack = 0;
00446             if (Velocity.X == 0) {
00447                 AnimationPlayer.Play("idle");
00448             } else {
00449                 AnimationPlayer.Play("run");
00450             }
00451         } else if (!IsOnFloor() && !IsAttacking() && !IsBlocking()) {
00452             LastAttack = 0;
00453             if (Velocity.Y < 0) {
00454                 AnimationPlayer.Play("jump");
00455             } else if (Velocity.Y > 0) {
00456                 AnimationPlayer.Play("fall");
00457             }
00458         }
00459     }
00460 }
```

9.45 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/scripts/PlayerStats.cs File Reference

Classes

- class PlayerStats

  Klasse für die Spielerstats.

9.46 PlayerStats.cs

Go to the documentation of this file.

```
00001 using System;
00002 using Godot;
00003
00007 public partial class PlayerStats : Node
00008 {
00009
00010     public static PlayerStats Instance { get; private set; }
00011
00012     private String RespawnLevelTag = "intro";
00013     private String CurrentLevelTag = "intro";
00014     private Vector2 SpawnPoint;
00015     private Vector2 Position = new Vector2(-540, 160);
00016     private int SinAmount;
00017     private float MaxHealthPoints = 100f;
00018     private float CurrentHealth;
00019     private float MaxStamina = 100f;
00020     private float CurrentStamina;
00021     private int BVHealAmount = 25;
00022     private int BVMaxUses = 5;
00023     private int BVCurrentUses;
00024
00025
00029     public override void _Ready(){
00030         CurrentHealth = MaxHealthPoints;
00031         CurrentStamina = MaxStamina;
00032         BVCurrentUses = BVMaxUses;
00033         Instance = this;
00034     }
00035
00040     public String GetRespawnLevelTag() {
00041         return RespawnLevelTag;
00042     }
00043
00048     public void SetRespawnLevelTag(String levelTag) {
00049         RespawnLevelTag = levelTag;
00050     }
00051
00056     public String GetCurrentLevelTag() {
00057         return CurrentLevelTag;
00058     }
00059
00064     public void SetCurrentLevelTag(String levelTag) {
00065         CurrentLevelTag = levelTag;
00066     }
00067
00072     public void SetSpawnPoint(Vector2 spawnPoint) {
00073         SpawnPoint = spawnPoint;
00074     }
00075
00080     public Vector2 GetSpawnPoint(){
00081         return SpawnPoint;
00082     }
00083
00088     public void SetPosition(Vector2 position) {
00089         Position = position;
00090     }
00091
00096     public Vector2 GetPosition(){
00097         return Position;
00098     }
00099
00100
00105     public int GetSinAmount(){
00106         return SinAmount;
00107     }
00108
00113     public void SetSinAmount(int Value) {
00114         // SinAmount muss immer >= 0 sein
00115         SinAmount = Mathf.Max(Value, 0);
00116     }
00117
00122     public float GetMaxHealthPoints(){
00123         return MaxHealthPoints;
00124     }
00125
00130     public void SetMaxHealthPoints(float maxHealthPoints){
00131         // MaxHealthPoints muss immer positiv sein
00132         MaxHealthPoints = Mathf.Max(maxHealthPoints, 1); // Verhindert, dass MaxHealthPoints <= 0 wird
00133     }
00134
00139     public float GetCurrentHealth(){
00140         return CurrentHealth;
```

```
00141     }
00142
00147     public void SetCurrentHealth(float Health){
00148         // CurrentHealth darf MaxHealthPoints nicht überschreiten.
00149         CurrentHealth = Mathf.Min(Health, MaxHealthPoints);
00150     }
00151
00156     public void SetMaxStamina(float Value) {
00157         // MaxStamina muss immer positiv sein
00158         MaxStamina = Mathf.Max(Value, 1);
00159     }
00160
00165     public float GetMaxStamina() {
00166         return MaxStamina;
00167     }
00168
00173     public void SetStamina(float Value) {
00174         // Stellt sicher, dass die CurrentStamina im gültigen Bereich bleibt (zwischen 0 und
      MaxStamina)
00175         CurrentStamina = Mathf.Clamp(Value, 0, MaxStamina);
00176     }
00177
00182     public float GetStamina() {
00183         return CurrentStamina;
00184     }
00185
00190     public void SetBVHealAmount(int Value){
00191         BVHealAmount = Math.Max(0, Value);
00192     }
00193
00198     public int GetBVHealAmount() {
00199         return BVHealAmount;
00200     }
00201
00206     public void SetBVMaxUses(int Value){
00207         BVMaxUses = Math.Max(0, Value);
00208     }
00209
00214     public int GetBVMaxUses() {
00215         return BVMaxUses;
00216     }
00217
00222     public void SetBVCurrentUses(int Value){
00223         BVCurrentUses = Math.Max(0, Value);
00224     }
00225
00230     public int GetBVCurrentUses() {
00231         return BVCurrentUses;
00232     }
00233
00237     public void Reload(){
00238         Instance = new PlayerStats();
00239         Instance._Ready();
00240     }
00241
00242 }
```

9.47 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/scripts/Spike.cs File Reference

Classes

- class Spike

  Klasse für die Spikes.

9.48 Spike.cs

Go to the documentation of this file.

```
00001 using Godot;
00002 using System;
00003
00007 public partial class Spike : Node2D
```

```
00008 {
00009     // Variable für Player
00010     private Player Player;
00011
00012
00013     [Export]
00014     private float Damage = 10f;
00015
00020     public override void _Ready()
00021     {
00022         // Zugriff auf Player Node
00023
00024         Player = GetNode<Player>("../../Player");
00025     }
00026
00030     private void OnPlayerBodyEntered(Node body)
00031     {
00032
00033         if (body is Player)
00034         {
00035             Player = (Player)body; // Instanzvariable setzen
00036             Player.TakeDamage(GetDamage());
00037             Player.SlowPlayer(0.5f);
00038             GetNode<Timer>("StaticBody2D/Area2D/Timer").Start();
00039             GD.Print("Player entered spike");
00040         }
00041
00042
00043     }
00044
00048     private void OnPlayerBodyExited(Node body)
00049     {
00050         if (body is Player)
00051         {
00052             Player = null; // Instanzvariable zurücksetzen
00053             GetNode<Timer>("StaticBody2D/Area2D/Timer").Stop();
00054         }
00055     }
00056
00060     private void OnTimerTimeout()
00061     {
00062         GD.Print("Timer timeout");
00063         Player.TakeDamage(GetDamage());
00064         GetNode<Timer>("StaticBody2D/Area2D/Timer").Start();
00065     }
00066
00071     public Damage GetDamage()
00072     {
00073         return new Damage(0, Damage, Vector2.Zero, this);
00074     }
00075 }
```

9.49 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/scripts/SpikeDynamic.cs File Reference

Classes

- class SpikeDynamic

  Klasse für die beweglichen Spikes.

9.50 SpikeDynamic.cs

Go to the documentation of this file.

```
00001 using Godot;
00002 using System;
00003
00007 public partial class SpikeDynamic : Node2D
00008 {
00009     // Variable für Player
00010     private Player Player;
00011
00012     [Export]
```

```
00013     private float Damage = 10f;
00014
00020     public override void _Ready()
00021     {
00022         // Zugriff auf Player Node
00023
00024         Player = GetNode<Player>("../../../Player");
00025     }
00026
00030     private void OnPlayerBodyEntered(Node body)
00031     {
00032
00033         if (body is Player)
00034         {
00035             Player = (Player)body; // Instanzvariable setzen
00036             Player.TakeDamage(GetDamage());
00037             Player.SlowPlayer(0.5f);
00038             GetNode<Timer>("StaticBody2D/Area2D/Timer").Start();
00039             GD.Print("Player entered spike");
00040         }
00041
00042
00043     }
00044
00048     private void OnPlayerBodyExited(Node body)
00049     {
00050         if (body is Player)
00051         {
00052             Player = null; // Instanzvariable zurücksetzen
00053             GetNode<Timer>("StaticBody2D/Area2D/Timer").Stop();
00054         }
00055     }
00056
00060     private void OnTimerTimeout()
00061     {
00062         GD.Print("Timer timeout");
00063         Player.TakeDamage(GetDamage());
00064         GetNode<Timer>("StaticBody2D/Area2D/Timer").Start();
00065     }
00066
00071     public Damage GetDamage()
00072     {
00073         return new Damage(0, Damage, Vector2.Zero, this);
00074     }
00075 }
```

9.51 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/StaminaBar.cs File Reference

Classes

- class StaminaBar

  Klasse für die Ausdauerleiste des Spielers. Synchronisiert die Anzeige der StaminaBar mit der Ausdauer des Spielers.

9.52 StaminaBar.cs

Go to the documentation of this file.

```
00001 using Godot;
00002
00007 public partial class StaminaBar : TextureProgressBar {
00008
00013     public override void _Ready() {
00014         // Setze die maximale Ausdauer der StaminaBar basierend auf der Spieler-MaxStamina
00015         MaxValue = PlayerStats.Instance.GetMaxStamina();
00016         Value = PlayerStats.Instance.GetStamina();
00017     }
00018
00024     public override void _Process(double DeltaTime) {
00025         // Aktualisiere den Wert der StaminaBar basierend auf der aktuellen Ausdauer des Spielers
00026         Value = PlayerStats.Instance.GetStamina();
00027     }
00028 }
```

9.53 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/Das↩ Spiel/anfaengerpraktikum/scripts/StorageManager.cs File Reference

Classes

- class StorageManager

  Klasse für das Speichern und Laden von Daten.

9.54 StorageManager.cs

Go to the documentation of this file.

```
00001 using Godot;
00002 using System;
00003 using System.Collections;
00004
00008 public partial class StorageManager : Node {
00009
00010     public static StorageManager Instance { get; private set; }
00011     private const String PathSettings = "user://settings.txt";
00012     private String[] PathSave = {"user://save1.dat", "user://save2.dat", "user://save3.dat"};
00013     private int LastSaveId = -1;
00014     private int Saves = 0;
00015
00016
00020     public override void _Ready(){
00021         LoadSettings();
00022         Instance = this;
00023     }
00024
00028     public void LoadSettings(){
00029         if(!FileAccess.FileExists(PathSettings)){
00030             return;
00031         }
00032         FileAccess File = FileAccess.Open(PathSettings, FileAccess.ModeFlags.Read);
00033         Saves = (int) File.GetVar();
00034         LastSaveId = (int) File.GetVar();
00035
00036         File.Close();
00037     }
00038
00043     public void LoadGameFile(int id){
00044         if(!FileAccess.FileExists(PathSave[id])){
00045             return;
00046         }
00047         FileAccess File = FileAccess.Open(PathSave[id], FileAccess.ModeFlags.Read);
00048         PlayerStats.Instance.SetRespawnLevelTag((String) File.GetVar());
00049         PlayerStats.Instance.SetCurrentLevelTag((String) File.GetVar());
00050         PlayerStats.Instance.SetSpawnPoint((Vector2) File.GetVar());
00051         PlayerStats.Instance.SetPosition((Vector2) File.GetVar());
00052         PlayerStats.Instance.SetSinAmount((int) File.GetVar());
00053         PlayerStats.Instance.SetMaxHealthPoints((float) File.GetVar());
00054         PlayerStats.Instance.SetCurrentHealth((float) File.GetVar());
00055         PlayerStats.Instance.SetMaxStamina((float) File.GetVar());
00056         PlayerStats.Instance.SetStamina((float) File.GetVar());
00057         PlayerStats.Instance.SetBVHealAmount((int) File.GetVar());
00058         PlayerStats.Instance.SetBVMaxUses((int) File.GetVar());
00059         PlayerStats.Instance.SetBVCurrentUses((int) File.GetVar());
00060
00061         File.Close();
00062     }
00063
00068     public void SaveAll(int id){
00069         SaveGameFile(id);
00070         SaveSettings();
00071     }
00072
00076     public void SaveSettings(){
00077         FileAccess File = FileAccess.Open(PathSettings, FileAccess.ModeFlags.Write);
00078         File.StoreVar(Saves);
00079         File.StoreVar(LastSaveId);
00080
00081         File.Close();
00082     }
00083
```

```
00088     public void SaveGameFile(int id){
00089         FileAccess File = FileAccess.Open(PathSave[id], FileAccess.ModeFlags.Write);
00090         File.StoreVar(PlayerStats.Instance.GetRespawnLevelTag());
00091         File.StoreVar(PlayerStats.Instance.GetCurrentLevelTag());
00092         File.StoreVar(PlayerStats.Instance.GetSpawnPoint());
00093         File.StoreVar(PlayerStats.Instance.GetPosition());
00094         File.StoreVar(PlayerStats.Instance.GetSinAmount());
00095         File.StoreVar(PlayerStats.Instance.GetMaxHealthPoints());
00096         File.StoreVar(PlayerStats.Instance.GetCurrentHealth());
00097         File.StoreVar(PlayerStats.Instance.GetMaxStamina());
00098         File.StoreVar(PlayerStats.Instance.GetStamina());
00099         File.StoreVar(PlayerStats.Instance.GetBVHealAmount());
00100         File.StoreVar(PlayerStats.Instance.GetBVMaxUses());
00101         File.StoreVar(PlayerStats.Instance.GetBVCurrentUses());
00102
00103         File.Close();
00104     }
00105
00110     public void SetLastSaveId(int id){
00111         LastSaveId = id;
00112     }
00113
00118     public int GetLastSaveId(){
00119         return LastSaveId;
00120     }
00121
00126     public void SetSaves(int Saves){
00127         this.Saves = Saves;
00128     }
00129
00134     public int GetSaves(){
00135         return Saves;
00136     }
00137 }
```

9.55 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/TestClass.cs File Reference

Classes

- class **GdMUT.TestClass**

  This is a test class for GDMUT. This is purely for demonstration. If you added this into your project, feel free to delete it =).

Namespaces

- namespace GdMUT

9.56 C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/TestClass.cs

Go to the documentation of this file.

```
00001 namespace GdMUT;
00002
00007 public static class TestClass
00008 {
00009 #if TOOLS
00014     [CSTestFunction]
00015     public static Result ExamplePass()
00016     {
00017         int x = 0;
00018         x *= 100;
00019         return (x == 0) ? Result.Success : Result.Failure;
00020     }
00021
00026     [CSTestFunction]
00027     public static Result ExampleFail()
```

```
00028     {
00029         int x = 0;
00030         x *= 100;
00031         return (x != 0) ? Result.Success : Result.Failure;
00032     }
00033
00038     [CSTestFunction]
00039     public static Result ExampleCustomFail()
00040     {
00041         int x = 0;
00042         x *= 100;
00043         return (x != 0)
00044             ? Result.Success
00045             : new Result(false, "You can't multiply 0 and expect anything else than 0!");
00046     }
00047
00052     [CSTestFunction]
00053     public static Result ExampleCustomSuccess()
00054     {
00055         int x = 0;
00056         x *= 100;
00057         return (x == 0) ? new Result(true, "Proved that 0 * 100 = 0") : Result.Failure;
00058     }
00059 #endif
00060 }
```

Index

_PhysicsProcess
    Player, 88
_Process
    BaseEnemy, 33
    Boss1, 52
    HealthBar, 63
    Hud, 64
    Interactable, 68
    MainMenuBackground, 80
    StaminaBar, 120
_Ready
    BaseEnemy, 34
    BloodVial, 48
    Boss1, 52
    Checkpoint, 56
    Door, 61
    HealthBar, 63
    Hud, 65
    Interactable, 68
    LevelManager, 71
    MainMenu, 73
    NavigationManager, 82
    Player, 88
    PlayerStats, 103
    Spike, 114
    SpikeDynamic, 117
    StaminaBar, 120
    StorageManager, 122

AddMaxUses
    BloodVial, 48
AlreadyHit
    BaseEnemy, 42
anfaengerpraktikum, 1
AnimationPlayer
    Hud, 66
    Player, 97
AnimationState
    BaseEnemy, 42
Area
    Interactable, 69
Armor
    BaseEnemy, 43
ATTACK
    BaseEnemy, 33

BaseEnemy, 31
    _Process, 33
    _Ready, 34
    AlreadyHit, 42
    AnimationState, 42
    Armor, 43
    ATTACK, 33
    CheckLineOfSight, 34
    CheckPlayerHit, 35
    CollisionPolygon, 43
    CurrentHealthPoints, 43
    CurrentStamina, 43
    CurrentTarget, 43
    Damage, 43
    Dead, 44
    Die, 35
    FlipRotation, 36
    FrontCollisionRayCast, 44
    HandleMovement, 36
    HealthBar, 44
    Id, 47
    IDLE, 33
    IsCloseTo, 37
    IsDead, 39
    LeftFallProtection, 44
    LineOfSight, 44
    MainCollision, 44
    MaxHealthPoints, 45
    MaxStamina, 45
    OnDetectionBodyEntered, 39
    OnHitboxAreaEntered, 39
    OnPursuingRadiusBodyExited, 40
    OnSwordHitBoxBodyEntered, 40
    Player, 45
    Pursuing, 45
    Respawn, 40
    Respawnable, 45
    ReturnToStart, 45
    ReturnToStartAfter, 46
    RightFallProtection, 46
    SetRotation, 41
    SinAmount, 46
    Speed, 46
    Sprite, 46
    StartPosition, 46
    StartRotation, 47
    State, 33
    SwordHitbox, 47
    TAKE_HIT, 33
    TakeDamage, 41
    TargetPosition, 47
    UpdateAnimation, 42
    WALK, 33

BloodVial, 48
_Ready, 48
AddMaxUses, 48
LevelHealAmount, 49
ResetUses, 49
UseBloodVial, 49
BloodVials
Player, 97
Boss1, 50
_Process, 52
_Ready, 52
EnemiesRevived, 54
HandleRegeneration, 52
RegenAmount, 54
RegenCooldown, 55
RegenTimer, 55
ReviveEnemies, 53
ShowPopupMessage, 53
StartGlowing, 54
Buttons
Hud, 66
BVCurrentUses
PlayerStats, 111
BVHealAmount
PlayerStats, 111
BVMaxUses
PlayerStats, 111

C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/addons/GDMUT/Dock.cs, 127
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/addons/GDMUT/GDMUT.cs, 129
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/addons/GDMUT/MethodResult.cs, 130
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/addons/GDMUT/Result.cs, 131
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/addons/GDMUT/TestFunction.cs, 131
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/addons/GDMUT/TestLoader.cs, 131
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/addons/GDMUT/TestResult.cs, 133
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/addons/godot-git-plugin/THIRDPARTY.md, 134
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/README.md, 134
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/BaseEnemy.cs, 134
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/BloodVial.cs, 138
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Boss1.cs, 139
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Checkpoint.cs, 140
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Damage.cs, 141
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Door.cs, 141, 142
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Dummy.cs, 142
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Enemy1.cs, 143
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Enemy2.cs, 143, 144
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Enemy3.cs, 144
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Enemy4.cs, 145
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Hud.cs, 147
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/MainMenu.cs, 148
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Player.cs, 149
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/PlayerStats.cs, 153, 154
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Projectile.cs, 155
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/SaveManager.cs, 156
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/SceneTransitionManager.cs, 157
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Stamina.cs, 158
C:/Users/Youssef/Desktop/UNI/S4/ComputerspielEntwicklung/DasSpiel/anfaengerpraktikum/scripts/Test1.cs, 159
CanDash
Player, 97
Change
MainMenu, 73
CheckLineOfSight
BaseEnemy, 34
CheckPlayerHit
BaseEnemy, 35
Checkpoint, 58
_Ready, 56
OnPlayerBodyEntered, 56
Player, 56
CollisionPolygon
BaseEnemy, 43
ContinueButton
MainMenu, 78
CreateDashEffect
Player, 88
CurrentHealth
PlayerStats, 112
CurrentHealthPoints
BaseEnemy, 43
CurrentLevelTag
PlayerStats, 112
CurrentStamina
BaseEnemy, 43
PlayerStats, 112
CurrentTarget
BaseEnemy, 43

Damage, 57

BaseEnemy, 43
Damage, 57
GetPhysicalDMG, 58
GetPushAmount, 58
GetSource, 58
GetTrueDMG, 58
PhysicalDMG, 59
PushAmount, 59
Source, 59
Spike, 116
SpikeDynamic, 119
TrueDMG, 59
DashDirection
Player, 98
DashEffect
Player, 98
DashInProgress
Player, 89
DashSpeed
Player, 98
DashTimer
Player, 98
DashTrailInterval
Player, 98
DashTrailTimer
Player, 98
Dead
BaseEnemy, 44
DeferredChangeScene
NavigationManager, 82
DeleteButton
MainMenu, 78
DeleteConfirmation
MainMenu, 78
DestinationDoorTag
Door, 62
DestinationLevelTag
Door, 62
Die
BaseEnemy, 35
Door, 60
_Ready, 61
DestinationDoorTag, 62
DestinationLevelTag, 62
OnPlayerBodyEntered, 61
Spawn, 61
SpawnDirection, 62

Enabled
Hud, 67
EnemiesRevived
Boss1, 54

FlipRotation
BaseEnemy, 36
FrontCollisionRayCast
BaseEnemy, 44

GdMUT, 29
GdMUT.Components, 29
GetBloodVials
Player, 90
GetBVCurrentUses
PlayerStats, 103
GetBVHealAmount
PlayerStats, 103
GetBVMaxUses
PlayerStats, 103
GetCurrentHealth
PlayerStats, 104
GetCurrentLevelTag
PlayerStats, 104
GetDamage
Player, 90
Spike, 114
SpikeDynamic, 117
GetLastSaveId
StorageManager, 122
GetMaxHealthPoints
PlayerStats, 104
GetMaxStamina
PlayerStats, 105
GetPhysicalDMG
Damage, 58
GetPosition
PlayerStats, 105
GetPushAmount
Damage, 58
GetRespawnLevelTag
PlayerStats, 105
GetSaves
StorageManager, 122
GetSinAmount
PlayerStats, 106
GetSource
Damage, 58
GetSpawnPoint
PlayerStats, 106
GetStamina
PlayerStats, 106
GetTrueDMG
Damage, 58
GoToLevel
NavigationManager, 83

HandleJump
Player, 90
HandleMovement
BaseEnemy, 36
Player, 91
HandleRegeneration
Boss1, 52
HauptHitbox
Player, 99
HealthBar, 62
_Process, 63
_Ready, 63
BaseEnemy, 44

Hud, 64
_Process, 64
_Ready, 65
AnimationPlayer, 66
Buttons, 66
Enabled, 67
OnResumeButtonPressed, 65
OnSaveButtonPressed, 65
OnSaveMenuButtonPressed, 65
OnSaveQuitButtonPressed, 66
TogglePause, 66

Id
BaseEnemy, 47
IDLE
BaseEnemy, 33
InfoLabel
MainMenu, 79
Instance
NavigationManager, 85
PlayerStats, 113
StorageManager, 126
Interactable, 67
_Process, 68
_Ready, 68
Area, 69
OnAreaBodyExited, 68
Player, 69
PopUp, 69
Text, 70
TextLabel, 69
IsAttacking
Player, 92
IsBlocking
Player, 92
IsCloseTo
BaseEnemy, 37
IsDashing
Player, 99
IsDead
BaseEnemy, 39

JUMP_VELOCITY
Player, 99
JumpCount
Player, 99
JumpMax
Player, 99

LastAttack
Player, 99
LastSaveId
StorageManager, 125
LeftFallProtection
BaseEnemy, 44
LevelHealAmount
BloodVial, 49
LevelManager, 70
_Ready, 71
OnLevelSpawn, 71
LineOfSight
BaseEnemy, 44
LoadGameFile
StorageManager, 122
LoadSettings
StorageManager, 123

MainCollision
BaseEnemy, 44
MainMenu, 72
_Ready, 73
Change, 73
ContinueButton, 78
DeleteButton, 78
DeleteConfirmation, 78
InfoLabel, 79
MenuState, 79
Navigation, 79
OnBackButtonPressed, 74
OnContinueButtonPressed, 74
OnDeleteConfirmationCanceled, 75
OnDeleteConfirmationCloseRequested, 75
OnDeleteConfirmationConfirmed, 75
OnLoadGameButtonPressed, 75
OnNewGameButtonPressed, 76
OnQuitButtonPressed, 76
OnSave1DeletePressed, 76
OnSave1SelectPressed, 76
OnSave2DeletePressed, 77
OnSave2SelectPressed, 77
OnSave3DeletePressed, 77
OnSave3SelectPressed, 78
SaveLabel, 79
SavesContainer, 79
SaveToDelete, 79
SelectButton, 80
MainMenuBackground, 80
_Process, 80
ScrollSpeed, 81
MaxHeal
Player, 92
MaxHealthPoints
BaseEnemy, 45
PlayerStats, 112
MaxStamina
BaseEnemy, 45
PlayerStats, 112
MenuState
MainMenu, 79

Navigation
MainMenu, 79
NavigationManager, 81
_Ready, 82
DeferredChangeScene, 82
GoToLevel, 83
Instance, 85
OnTriggerPlayerSpawnEventHandler, 83

SceneBoss, 84
SceneIntro, 84
SceneLevel1, 84
SceneLevelOne, 85
SceneLevelTwo, 85
SceneMainMenu, 85
SpawnDoorTag, 85
TriggerPlayerSpawn, 84

OnAreaBodyExited
Interactable, 68
OnBackButtonPressed
MainMenu, 74
OnContinueButtonPressed
MainMenu, 74
OnDeleteConfirmationCanceled
MainMenu, 75
OnDeleteConfirmationCloseRequested
MainMenu, 75
OnDeleteConfirmationConfirmed
MainMenu, 75
OnDetectionBodyEntered
BaseEnemy, 39
OnHitboxAreaEntered
BaseEnemy, 39
OnLevelSpawn
LevelManager, 71
OnLoadGameButtonPressed
MainMenu, 75
OnNewGameButtonPressed
MainMenu, 76
OnPlayerBodyEntered
Checkpoint, 56
Door, 61
Spike, 115
SpikeDynamic, 118
OnPlayerBodyExited
Spike, 115
SpikeDynamic, 118
OnPursuingRadiusBodyExited
BaseEnemy, 40
OnQuitButtonPressed
MainMenu, 76
OnResumeButtonPressed
Hud, 65
OnSave1DeletePressed
MainMenu, 76
OnSave1SelectPressed
MainMenu, 76
OnSave2DeletePressed
MainMenu, 77
OnSave2SelectPressed
MainMenu, 77
OnSave3DeletePressed
MainMenu, 77
OnSave3SelectPressed
MainMenu, 78
OnSaveButtonPressed
Hud, 65
OnSaveMenuButtonPressed
Hud, 65
OnSaveQuitButtonPressed
Hud, 66
OnSpawn
Player, 93
OnSwordHitBoxBodyEntered
BaseEnemy, 40
OnTimerTimeout
Spike, 115
SpikeDynamic, 118
OnTriggerPlayerSpawnEventHandler
NavigationManager, 83

PathSave
StorageManager, 125
PathSettings
StorageManager, 126
PhysicalDMG
Damage, 59
Player, 86
_PhysicsProcess, 88
_Ready, 88
AnimationPlayer, 97
BaseEnemy, 45
BloodVials, 97
CanDash, 97
Checkpoint, 56
CreateDashEffect, 88
DashDirection, 98
DashEffect, 98
DashInProgress, 89
DashSpeed, 98
DashTimer, 98
DashTrailInterval, 98
DashTrailTimer, 98
GetBloodVials, 90
GetDamage, 90
HandleJump, 90
HandleMovement, 91
HauptHitbox, 99
Interactable, 69
IsAttacking, 92
IsBlocking, 92
IsDashing, 99
JUMP_VELOCITY, 99
JumpCount, 99
JumpMax, 99
LastAttack, 99
MaxHeal, 92
OnSpawn, 93
PlayerHitbox, 100
RegenerateStamina, 93
Respawn, 94
SetSinAmount, 94
SinDisplay, 100
SlowPlayer, 94
SPEED, 100
Spike, 116

SpikeDynamic, 119
Sprite, 100
StartDash, 95
StopDash, 95
SwordCollision, 100
TakeDamage, 95
TimeSinceLastStaminaUse, 100
UpdateAnimations, 96
UseStamina, 97
PlayerHitbox
Player, 100
PlayerStats, 101
_Ready, 103
BVCurrentUses, 111
BVHealAmount, 111
BVMaxUses, 111
CurrentHealth, 112
CurrentLevelTag, 112
CurrentStamina, 112
GetBVCurrentUses, 103
GetBVHealAmount, 103
GetBVMaxUses, 103
GetCurrentHealth, 104
GetCurrentLevelTag, 104
GetMaxHealthPoints, 104
GetMaxStamina, 105
GetPosition, 105
GetRespawnLevelTag, 105
GetSinAmount, 106
GetSpawnPoint, 106
GetStamina, 106
Instance, 113
MaxHealthPoints, 112
MaxStamina, 112
Position, 112
Reload, 107
RespawnLevelTag, 113
SetBVCurrentUses, 107
SetBVHealAmount, 107
SetBVMaxUses, 108
SetCurrentHealth, 108
SetCurrentLevelTag, 108
SetMaxHealthPoints, 109
SetMaxStamina, 109
SetPosition, 109
SetRespawnLevelTag, 110
SetSinAmount, 110
SetSpawnPoint, 110
SetStamina, 111
SinAmount, 113
SpawnPoint, 113
PopUp
Interactable, 69
Position
PlayerStats, 112
Pursuing
BaseEnemy, 45
PushAmount
Damage, 59

RegenAmount
Boss1, 54
RegenCooldown
Boss1, 55
RegenerateStamina
Player, 93
RegenTimer
Boss1, 55
Reload
PlayerStats, 107
ResetUses
BloodVial, 49
Respawn
BaseEnemy, 40
Player, 94
Respawnable
BaseEnemy, 45
RespawnLevelTag
PlayerStats, 113
ReturnToStart
BaseEnemy, 45
ReturnToStartAfter
BaseEnemy, 46
ReviveEnemies
Boss1, 53
RightFallProtection
BaseEnemy, 46

SaveAll
StorageManager, 123
SaveGameFile
StorageManager, 124
SaveLabel
MainMenu, 79
Saves
StorageManager, 126
SavesContainer
MainMenu, 79
SaveSettings
StorageManager, 124
SaveToDelete
MainMenu, 79
SceneBoss
NavigationManager, 84
SceneIntro
NavigationManager, 84
SceneLevel1
NavigationManager, 84
SceneLevelOne
NavigationManager, 85
SceneLevelTwo
NavigationManager, 85
SceneMainMenu
NavigationManager, 85
ScrollSpeed
MainMenuBackground, 81
SelectButton

MainMenu, 80
SetBVCurrentUses
PlayerStats, 107
SetBVHealAmount
PlayerStats, 107
SetBVMaxUses
PlayerStats, 108
SetCurrentHealth
PlayerStats, 108
SetCurrentLevelTag
PlayerStats, 108
SetLastSaveId
StorageManager, 124
SetMaxHealthPoints
PlayerStats, 109
SetMaxStamina
PlayerStats, 109
SetPosition
PlayerStats, 109
SetRespawnLevelTag
PlayerStats, 110
SetRotation
BaseEnemy, 41
SetSaves
StorageManager, 125
SetSinAmount
Player, 94
PlayerStats, 110
SetSpawnPoint
PlayerStats, 110
SetStamina
PlayerStats, 111
ShowPopupMessage
Boss1, 53
SinAmount
BaseEnemy, 46
PlayerStats, 113
SinDisplay
Player, 100
SlowPlayer
Player, 94
Source
Damage, 59
Spawn
Door, 61
SpawnDirection
Door, 62
SpawnDoorTag
NavigationManager, 85
SpawnPoint
PlayerStats, 113
SPEED
Player, 100
Speed
BaseEnemy, 46
Spike, 114
_Ready, 114
Damage, 116
GetDamage, 114
OnPlayerBodyEntered, 115
OnPlayerBodyExited, 115
OnTimerTimeout, 115
Player, 116
SpikeDynamic, 116
_Ready, 117
Damage, 119
GetDamage, 117
OnPlayerBodyEntered, 118
OnPlayerBodyExited, 118
OnTimerTimeout, 118
Player, 119
Sprite
BaseEnemy, 46
Player, 100
StaminaBar, 119
_Process, 120
_Ready, 120
StartDash
Player, 95
StartGlowing
Boss1, 54
StartPosition
BaseEnemy, 46
StartRotation
BaseEnemy, 47
State
BaseEnemy, 33
StopDash
Player, 95
StorageManager, 121
_Ready, 122
GetLastSaveId, 122
GetSaves, 122
Instance, 126
LastSaveId, 125
LoadGameFile, 122
LoadSettings, 123
PathSave, 125
PathSettings, 126
SaveAll, 123
SaveGameFile, 124
Saves, 126
SaveSettings, 124
SetLastSaveId, 124
SetSaves, 125
SwordCollision
Player, 100
SwordHitbox
BaseEnemy, 47

TAKE_HIT
BaseEnemy, 33
TakeDamage
BaseEnemy, 41
Player, 95
TargetPosition
BaseEnemy, 47

Text
    Interactable, 70
TextLabel
    Interactable, 69
Third-Party Notices, 3
TimeSinceLastStaminaUse
    Player, 100
TogglePause
    Hud, 66
TriggerPlayerSpawn
    NavigationManager, 84
TrueDMG
    Damage, 59

UpdateAnimation
    BaseEnemy, 42
UpdateAnimations
    Player, 96
UseBloodVial
    BloodVial, 49
UseStamina
    Player, 97

WALK
    BaseEnemy, 33