Vuji

Создано системой Doxygen 1.9.5

1 Алфавитный указатель пространств имен 1
1.1 Пространства имен . . . 1

2 Иерархический список классов 3
2.1 Иерархия классов . . . 3

3 Алфавитный указатель классов 7
3.1 Классы . . . 7

4 Список файлов 11
4.1 Файлы . . . 11

5 Пространства имен 13
5.1 Пространство имен GameSettings . . . 13
5.2 Пространство имен ServersInfo . . . 13
5.3 Пространство имен StructsRequest . . . 13
5.4 Пространство имен StructsResponse . . . 13

6 Классы 15
6.1 Класс AcceptFriendInvite . . . 15
6.1.1 Подробное описание . . . 15
6.1.2 Методы . . . 15
6.1.2.1 OnConnectedToMaster() . . . 16
6.1.2.2 OnJoinedRoom() . . . 16
6.1.2.3 StartAcceptInvite() . . . 16
6.1.3 Данные класса . . . 16
6.1.3.1 roomName . . . 16
6.2 Класс AgressionTrigger . . . 17
6.2.1 Подробное описание . . . 17
6.3 Класс AnimationPlayer . . . 17
6.3.1 Подробное описание . . . 18
6.3.2 Методы . . . 18
6.3.2.1 ChangePlayerAnimation() . . . 18
6.3.2.2 ChangePlayerAnimation_q() . . . 18
6.3.2.3 getCurrentMovingState() . . . 18
6.3.3 Данные класса . . . 19
6.3.3.1 _attack . . . 19
6.3.3.2 _down . . . 19
6.3.3.3 _drink . . . 19
6.3.3.4 _idle . . . 19
6.3.3.5 _left . . . 19
6.3.3.6 _move . . . 19
6.3.3.7 _right . . . 20
6.3.3.8 _shot . . . 20
6.3.3.9 _up . . . 20

6.3.3.10 movingState . . . 20
6.4 Структура PlayerAOE.AOEstruct . . . 20
6.4.1 Подробное описание . . . 20
6.4.2 Данные класса . . . 21
6.4.2.1 AOE . . . 21
6.4.2.2 AOEKey . . . 21
6.5 Класс Appe . . . 21
6.5.1 Подробное описание . . . 21
6.5.2 Методы . . . 21
6.5.2.1 UseItem() . . . 22
6.6 Класс AttackTrigger . . . 22
6.6.1 Подробное описание . . . 22
6.7 Класс BaseAOE . . . 22
6.7.1 Подробное описание . . . 23
6.7.2 Методы . . . 23
6.7.2.1 GetAoeDescription() . . . 23
6.7.2.2 GetAoeName() . . . 23
6.7.2.3 SetDamageTags() . . . 23
6.7.2.4 SetDestroyTags() . . . 24
6.7.2.5 SetSenderCollider() . . . 24
6.8 Класс BaseEffect . . . 24
6.8.1 Подробное описание . . . 25
6.8.2 Методы . . . 25
6.8.2.1 ApplyEffect() . . . 25
6.8.3 Данные класса . . . 25
6.8.3.1 description . . . 25
6.8.3.2 duration . . . 25
6.8.3.3 effectName . . . 25
6.8.3.4 effectSprite . . . 26
6.9 Класс BaseEntity . . . 26
6.9.1 Подробное описание . . . 27
6.9.2 Методы . . . 27
6.9.2.1 AddEffect() . . . 27
6.9.2.2 DecreaseDamage() . . . 27
6.9.2.3 DecreaseDefense() . . . 28
6.9.2.4 DecreaseSpeed() . . . 28
6.9.2.5 deSelectSkill() . . . 28
6.9.2.6 GetBaseDamage() . . . 28
6.9.2.7 GetDefense() . . . 28
6.9.2.8 GetEnergyPoints() . . . 29
6.9.2.9 GetEntityName() . . . 29
6.9.2.10 GetHealthPoints() . . . 29
6.9.2.11 GetMaxEnergyPoints() . . . 29

6.9.2.12 GetMaxHealthPoints() . . . 29
6.9.2.13 GetMoveSpeed() . . . 30
6.9.2.14 GetSelectedSkill() . . . 30
6.9.2.15 Heal() . . . 30
6.9.2.16 IncreaseDamage() . . . 30
6.9.2.17 IncreaseDefense() . . . 30
6.9.2.18 IncreaseMaxEnergy() . . . 31
6.9.2.19 IncreaseMaxHealth() . . . 31
6.9.2.20 IncreaseSpeed() . . . 31
6.9.2.21 selectSkill() . . . 31
6.9.2.22 setIsCooldown() . . . 32
6.9.2.23 spendEnergy() . . . 32
6.9.2.24 TakeDamage() . . . 32
6.9.2.25 TickPoints() . . . 32
6.9.2.26 UpdateText() . . . 33
6.9.2.27 UseSkill() . . . 33
6.9.3 Данные класса . . . 33
6.9.3.1 _controller . . . 33
6.9.3.2 _droppedItemPrefab . . . 33
6.9.3.3 displayedName . . . 34
6.9.3.4 healthBar . . . 34
6.9.3.5 OnEffectApply . . . 34
6.9.3.6 OnSkillSelectionChange . . . 34
6.9.3.7 skills . . . 34
6.9.3.8 teamSpawn . . . 34
6.9.4 Полный список свойств . . . 34
6.9.4.1 isDead . . . 35
6.10 Класс BaseItem . . . 35
6.10.1 Подробное описание . . . 35
6.10.2 Методы . . . 35
6.10.2.1 GetAmount() . . . 35
6.10.2.2 GetDescription() . . . 36
6.10.2.3 GetImage() . . . 36
6.10.2.4 GetItemName() . . . 36
6.10.2.5 SetAmount() . . . 36
6.10.2.6 UseItem() . . . 37
6.11 Класс BasePassiveSkill . . . 37
6.11.1 Подробное описание . . . 37
6.11.2 Методы . . . 37
6.11.2.1 ActivatePassiveSkill() . . . 38
6.11.2.2 GetDescription() . . . 38
6.11.2.3 GetName() . . . 38
6.11.3 Данные класса . . . 38

6.11.3.1 skillDescription . . . 38
6.11.3.2 skillName . . . 38
6.12 Класс BaseProjectile . . . 39
6.12.1 Подробное описание . . . 39
6.12.2 Методы . . . 39
6.12.2.1 AddDamage() . . . 39
6.12.2.2 GetProjectileDescription() . . . 39
6.12.2.3 GetProjectileName() . . . 40
6.12.2.4 SetAimAngel() . . . 40
6.12.2.5 SetAimDirection() . . . 40
6.12.2.6 SetDamageTags() . . . 40
6.12.2.7 SetDestroyTags() . . . 40
6.12.2.8 SetSenderCollider() . . . 41
6.13 Класс BaseSkill . . . 41
6.13.1 Подробное описание . . . 42
6.13.2 Методы . . . 42
6.13.2.1 GetCastTime() . . . 42
6.13.2.2 GetCooldownTime() . . . 42
6.13.2.3 GetCost() . . . 42
6.13.2.4 GetDescription() . . . 42
6.13.2.5 GetName() . . . 43
6.13.2.6 GetSprite() . . . 43
6.13.2.7 UseSkill() . . . 43
6.13.3 Данные класса . . . 43
6.13.3.1 cancelMovementOnCast . . . 43
6.13.3.2 castTime . . . 43
6.13.3.3 cooldown . . . 44
6.13.3.4 energyCost . . . 44
6.13.3.5 onCast . . . 44
6.13.3.6 onRelease . . . 44
6.13.3.7 skillDescription . . . 44
6.13.3.8 skillName . . . 44
6.13.3.9 skillSprite . . . 45
6.14 Класс BaseUpgrade . . . 45
6.14.1 Подробное описание . . . 45
6.14.2 Методы . . . 45
6.14.2.1 ApplyUpgrade() . . . 45
6.14.2.2 GetCost() . . . 46
6.14.2.3 GetDescription() . . . 46
6.14.2.4 GetName() . . . 46
6.15 Класс Berserk . . . 46
6.15.1 Подробное описание . . . 46
6.16 Класс CameraFollow . . . 47

6.16.1 Подробное описание . . . 47
6.16.2 Данные класса . . . 47
6.16.2.1 player . . . 47
6.17 Класс CameraPlayer . . . 47
6.17.1 Подробное описание . . . 48
6.18 Класс ClassSelection . . . 48
6.18.1 Подробное описание . . . 48
6.19 Класс Connect . . . 48
6.19.1 Подробное описание . . . 48
6.19.2 Методы . . . 49
6.19.2.1 OnConnectedToMaster() . . . 49
6.20 Класс Controllers . . . 49
6.20.1 Подробное описание . . . 50
6.20.2 Методы . . . 50
6.20.2.1 CheckVujiServer() . . . 50
6.20.2.2 FindFriendsByName() . . . 50
6.20.2.3 GetUserID() . . . 50
6.20.2.4 Login() . . . 51
6.20.2.5 Register() . . . 51
6.20.2.6 SetLocalUserName() . . . 52
6.20.2.7 UserOffline() . . . 52
6.20.2.8 UserOnline() . . . 52
6.21 Класс DataBase . . . 53
6.21.1 Подробное описание . . . 53
6.21.2 Методы . . . 53
6.21.2.1 AddKeybind() . . . 53
6.21.2.2 AddSetting() . . . 54
6.21.2.3 ExistKeybind() . . . 54
6.21.2.4 ExistSetting() . . . 55
6.21.2.5 GetKeybinds() . . . 55
6.21.2.6 GetSettings() . . . 56
6.21.2.7 GetToken() . . . 56
6.21.2.8 SetKeybind() . . . 57
6.21.2.9 SetSetting() . . . 57
6.21.2.10 SetToken() . . . 58
6.22 Класс DISABLESUKA . . . 58
6.22.1 Подробное описание . . . 59
6.23 Класс DisplayedItem . . . 59
6.23.1 Подробное описание . . . 59
6.23.2 Методы . . . 59
6.23.2.1 OnPointerDown() . . . 59
6.23.2.2 OnPointerUp() . . . 60
6.23.3 Данные класса . . . 60

6.23.3.1 displayedItem . . . 60
6.23.3.2 inventoryPanel . . . 60
6.23.3.3 itemId . . . 61
6.23.3.4 itemPosition . . . 61
6.23.3.5 onItemDrop . . . 61
6.23.3.6 onItemSwap . . . 61
6.24 Класс DroppedItem . . . 61
6.24.1 Подробное описание . . . 62
6.24.2 Методы . . . 62
6.24.2.1 SetItem() . . . 62
6.24.3 Данные класса . . . 62
6.24.3.1 itemData . . . 62
6.25 Класс Drunk . . . 62
6.25.1 Подробное описание . . . 63
6.25.2 Методы . . . 63
6.25.2.1 ApplyEffect() . . . 63
6.25.2.2 DrunkEffect() . . . 63
6.25.3 Данные класса . . . 63
6.25.3.1 additionalDefense . . . 64
6.26 Класс Drunkard . . . 64
6.26.1 Подробное описание . . . 64
6.26.2 Методы . . . 64
6.26.2.1 UseSkill() . . . 65
6.27 Класс EffectsListManager . . . 65
6.27.1 Подробное описание . . . 65
6.28 Класс EffectTimerManager . . . 66
6.28.1 Подробное описание . . . 66
6.28.2 Методы . . . 66
6.28.2.1 SetEffect() . . . 66
6.29 Класс ENABLESUKA . . . 67
6.29.1 Подробное описание . . . 67
6.30 Класс EndGame . . . 67
6.30.1 Подробное описание . . . 67
6.30.2 Методы . . . 68
6.30.2.1 TeamOneWin() . . . 68
6.30.2.2 TeamTwoWin() . . . 68
6.30.3 Данные класса . . . 68
6.30.3.1 OnGameEnd . . . 68
6.31 Класс EndGameManager . . . 68
6.31.1 Подробное описание . . . 69
6.31.2 Методы . . . 69
6.31.2.1 LeaveGame() . . . 69
6.32 Класс EnemyAI . . . 69

6.32.1 Подробное описание . . . 70
6.32.2 Методы . . . 70
6.32.2.1 AgressionStart() . . . 70
6.32.3 Данные класса . . . 70
6.32.3.1 reachedEndOfPath . . . 70
6.33 Класс EnergyBarUI . . . 70
6.33.1 Подробное описание . . . 71
6.33.2 Методы . . . 71
6.33.2.1 SetEntity() . . . 71
6.34 Класс EntityMelee . . . 71
6.34.1 Подробное описание . . . 71
6.34.2 Методы . . . 71
6.34.2.1 Attack() . . . 72
6.35 Класс EntityNameManager . . . 72
6.35.1 Подробное описание . . . 73
6.35.2 Методы . . . 73
6.35.2.1 GetOffset() . . . 73
6.35.2.2 SetOffset() . . . 73
6.35.3 Данные класса . . . 73
6.35.3.1 entityName . . . 73
6.36 Класс EscapeMenu . . . 74
6.36.1 Подробное описание . . . 74
6.36.2 Методы . . . 74
6.36.2.1 LeaveGame() . . . 74
6.36.2.2 LeaveLobby() . . . 75
6.36.2.3 OpenSettings() . . . 75
6.36.2.4 ResumeGame() . . . 75
6.37 Класс Exclamation . . . 75
6.37.1 Подробное описание . . . 76
6.37.2 Методы . . . 76
6.37.2.1 UseSkill() . . . 76
6.38 Класс StructsRequest.FindFriendsByNameStructRequest . . . 76
6.38.1 Подробное описание . . . 77
6.38.2 Данные класса . . . 77
6.38.2.1 friendsName . . . 77
6.39 Класс StructsResponse.FindFriendsByNameStructResponse . . . 77
6.39.1 Подробное описание . . . 77
6.39.2 Данные класса . . . 77
6.39.2.1 friends . . . 77
6.40 Класс FireballSkill . . . 78
6.40.1 Подробное описание . . . 78
6.40.2 Методы . . . 78
6.40.2.1 UseSkill() . . . 78

6.41 Класс FlurryOfFire . . . 79
6.41.1 Подробное описание . . . 79
6.41.2 Методы . . . 79
6.41.2.1 UseSkill() . . . 79
6.42 Класс FriendItemManager . . . 80
6.42.1 Подробное описание . . . 80
6.42.2 Методы . . . 80
6.42.2.1 InviteFriend() . . . 80
6.42.3 Данные класса . . . 81
6.42.3.1 lobbyManager . . . 81
6.42.3.2 userID . . . 81
6.42.3.3 usernameTextField . . . 81
6.43 Класс FriendsListController . . . 81
6.43.1 Подробное описание . . . 82
6.43.2 Методы . . . 82
6.43.2.1 FillFriendsList() . . . 82
6.43.2.2 FindFriendsByName() . . . 82
6.43.2.3 OpenFriendsList() . . . 83
6.44 Класс GameSettings.GameSettingsOriginal . . . 83
6.44.1 Подробное описание . . . 83
6.44.2 Данные класса . . . 83
6.44.2.1 MaxPlayersInGame . . . 83
6.45 Класс Generator . . . 84
6.45.1 Подробное описание . . . 84
6.45.2 Данные класса . . . 84
6.45.2.1 Height . . . 84
6.45.2.2 RoomPrefabs . . . 84
6.45.2.3 StartingRoom . . . 84
6.45.2.4 Width . . . 85
6.46 Класс HealthBarManager . . . 85
6.46.1 Подробное описание . . . 85
6.46.2 Методы . . . 85
6.46.2.1 GetOffset() . . . 86
6.46.2.2 SetHealth() . . . 86
6.46.2.3 SetOffset() . . . 86
6.46.3 Данные класса . . . 87
6.46.3.1 slider . . . 87
6.47 Класс HealthBarUI . . . 87
6.47.1 Подробное описание . . . 87
6.47.2 Методы . . . 87
6.47.2.1 SetEntity() . . . 87
6.48 Класс InputControllerPlayer . . . 88
6.48.1 Подробное описание . . . 88

6.49 Класс Inventory . . . 88
6.49.1 Подробное описание . . . 89
6.49.2 Методы . . . 89
6.49.2.1 AddItem() . . . 89
6.49.2.2 ClearInventory() . . . 89
6.49.2.3 DropItem() . . . 89
6.49.2.4 GetAllItems() . . . 90
6.49.2.5 SyncronisedDrop() . . . 90
6.49.3 Данные класса . . . 90
6.49.3.1 inventoryItems . . . 90
6.49.3.2 onItemAdded . . . 90
6.50 Класс InventoryWindow . . . 91
6.50.1 Подробное описание . . . 91
6.50.2 Методы . . . 91
6.50.2.1 OnSpawn() . . . 91
6.50.2.2 Start() . . . 91
6.51 Класс InviteFriend . . . 92
6.51.1 Подробное описание . . . 92
6.51.2 Методы . . . 92
6.51.2.1 OnJoinedRoom() . . . 92
6.51.2.2 StartInviteFriend() . . . 92
6.51.3 Данные класса . . . 93
6.51.3.1 invitedUserID . . . 93
6.51.3.2 roomName . . . 93
6.52 Класс Keybind . . . 93
6.52.1 Подробное описание . . . 93
6.52.2 Конструктор(ы) . . . 93
6.52.2.1 Keybind() . . . 93
6.52.3 Данные класса . . . 94
6.52.3.1 category . . . 94
6.52.3.2 key . . . 94
6.52.3.3 name . . . 94
6.53 Класс KeybindManager . . . 94
6.53.1 Подробное описание . . . 95
6.53.2 Методы . . . 95
6.53.2.1 GetKey() . . . 95
6.53.2.2 GetName() . . . 96
6.53.2.3 ResetKeybind() . . . 96
6.53.2.4 SetKey() . . . 96
6.53.2.5 SetKeybind() . . . 96
6.53.2.6 SetName() . . . 97
6.53.3 Данные класса . . . 97
6.53.3.1 Binding . . . 97

6.53.3.2 keyChanged . . . 97
6.54 Класс KeyHandler . . . 98
6.54.1 Подробное описание . . . 99
6.54.2 Методы . . . 99
6.54.2.1 GetKeybind() . . . 99
6.54.2.2 GetKeybinds() . . . 99
6.54.2.3 GetUIOpened() . . . 100
6.54.2.4 IsClearAndPlaying() . . . 100
6.54.2.5 IsPaused() . . . 100
6.54.2.6 NormalizeKeybind() . . . 100
6.54.2.7 Pause() . . . 101
6.54.2.8 SetKeybind() . . . 101
6.54.2.9 SetUIOpened() . . . 102
6.54.3 Данные класса . . . 102
6.54.3.1 abilityKeys . . . 102
6.54.3.2 AllKeys . . . 102
6.54.3.3 instance . . . 103
6.54.3.4 keyPressed . . . 103
6.54.3.5 movementKeys . . . 103
6.54.3.6 numbersKeyCodes . . . 103
6.54.3.7 uiKeys . . . 103
6.55 Класс LeaveRoom . . . 104
6.55.1 Подробное описание . . . 104
6.55.2 Методы . . . 104
6.55.2.1 HideLeaveRoomButton() . . . 104
6.55.2.2 LeaveFromRoom() . . . 104
6.55.2.3 ShowLeaveRoomButton() . . . 105
6.56 Класс LobbyManager . . . 105
6.56.1 Подробное описание . . . 105
6.56.2 Методы . . . 105
6.56.2.1 AcceptInviteFriend() . . . 105
6.56.2.2 CreateInviteFriend() . . . 106
6.56.2.3 CreateLobbyAndInviteUser() . . . 106
6.56.2.4 OnConnectedToMaster() . . . 107
6.56.2.5 OnJoinedRoom() . . . 107
6.56.2.6 ToggleSettings() . . . 107
6.56.3 Данные класса . . . 107
6.56.3.1 playerStatus . . . 107
6.57 Класс LoginManager . . . 108
6.57.1 Подробное описание . . . 108
6.57.2 Методы . . . 108
6.57.2.1 LoginInAccount() . . . 108
6.57.2.2 ShowRegisterScene() . . . 108

6.57.3 Данные класса . . . 109
6.57.3.1 loginInput . . . 109
6.57.3.2 passwordInput . . . 109
6.58 Класс StructsRequest.LoginStructRequest . . . 109
6.58.1 Подробное описание . . . 109
6.58.2 Данные класса . . . 109
6.58.2.1 login . . . 109
6.58.2.2 password . . . 110
6.59 Класс ServersInfo.MainServerInfo . . . 110
6.59.1 Подробное описание . . . 110
6.59.2 Данные класса . . . 110
6.59.2.1 ServerDomain . . . 110
6.60 Класс ManagerGame . . . 110
6.60.1 Подробное описание . . . 111
6.60.2 Методы . . . 111
6.60.2.1 OnPlayerPropertiesUpdate() . . . 111
6.61 Класс MovementPlayer . . . 111
6.61.1 Подробное описание . . . 112
6.61.2 Методы . . . 112
6.61.2.1 cancelMovement() . . . 112
6.61.3 Данные класса . . . 112
6.61.3.1 canMove . . . 112
6.62 Класс NoticeInviteManager . . . 112
6.62.1 Подробное описание . . . 113
6.62.2 Методы . . . 113
6.62.2.1 AcceptInvite() . . . 113
6.62.2.2 CancelInvite() . . . 113
6.62.3 Данные класса . . . 113
6.62.3.1 lobbyManager . . . 114
6.62.3.2 roomName . . . 114
6.62.3.3 usernameTextField . . . 114
6.63 Класс NoticeListController . . . 114
6.63.1 Подробное описание . . . 114
6.63.2 Методы . . . 114
6.63.2.1 AddInviteNotice() . . . 114
6.63.2.2 OpenNoticeList() . . . 115
6.64 Класс OnFire . . . 115
6.64.1 Подробное описание . . . 116
6.64.2 Методы . . . 116
6.64.2.1 ApplyEffect() . . . 116
6.64.2.2 OnFireEffect() . . . 116
6.65 Класс Play . . . 116
6.65.1 Подробное описание . . . 117

6.65.2 Методы . . . 117
6.65.2.1 OnConnectedToMaster() . . . 117
6.65.2.2 OnFriendListUpdate() . . . 117
6.65.2.3 StartPlayInGame() . . . 118
6.66 Класс PlayerAOE . . . 118
6.66.1 Подробное описание . . . 118
6.66.2 Методы . . . 119
6.66.2.1 Attack() . . . 119
6.67 Класс PlayerMelee . . . 119
6.67.1 Подробное описание . . . 119
6.67.2 Методы . . . 119
6.67.2.1 MasterCheckMeleeAttack() . . . 120
6.68 Класс PlayerProjectile . . . 120
6.68.1 Подробное описание . . . 120
6.68.2 Методы . . . 120
6.68.2.1 Attack() [1/2] . . . 121
6.68.2.2 Attack() [2/2] . . . 121
6.69 Класс PlayerScript . . . 121
6.69.1 Подробное описание . . . 121
6.69.2 Методы . . . 122
6.69.2.1 KillPlayer() . . . 122
6.70 Класс PlayersFounded . . . 122
6.70.1 Подробное описание . . . 122
6.70.2 Методы . . . 123
6.70.2.1 HidePlayersFounded() . . . 123
6.70.2.2 ShowPlayersFounded() . . . 123
6.70.2.3 UpdatePlayersFounded() . . . 123
6.71 Класс PlayersTeamsManager . . . 124
6.71.1 Подробное описание . . . 124
6.71.2 Методы . . . 124
6.71.2.1 PlayerInTeamOneDied() . . . 124
6.71.2.2 PlayerInTeamTwoDied() . . . 124
6.72 Класс PlayerUpgrades . . . 125
6.72.1 Подробное описание . . . 125
6.72.2 Методы . . . 125
6.72.2.1 AddUpgrade() . . . 125
6.72.2.2 AddXp() . . . 126
6.72.2.3 GetXp() . . . 126
6.72.2.4 SwitchPanels() . . . 126
6.73 Класс Poison . . . 126
6.73.1 Подробное описание . . . 127
6.73.2 Методы . . . 127
6.73.2.1 ApplyEffect() . . . 127

6.73.2.2 PoisonEffect() . . . 127
6.73.3 Данные класса . . . 127
6.73.3.1 damageTickSeconds . . . 128
6.73.3.2 posionDamage . . . 128
6.74 Структура PlayerProjectile.Projectile . . . 128
6.74.1 Подробное описание . . . 128
6.74.2 Данные класса . . . 128
6.74.2.1 projectile . . . 128
6.74.2.2 projectileKey . . . 128
6.75 Класс RegisterManager . . . 129
6.75.1 Подробное описание . . . 129
6.75.2 Методы . . . 129
6.75.2.1 RegisterAccount() . . . 129
6.75.2.2 ShowLoginScene() . . . 130
6.75.3 Данные класса . . . 130
6.75.3.1 loginInput . . . 130
6.75.3.2 passwordOneInput . . . 130
6.75.3.3 passwordTwoInput . . . 130
6.76 Класс StructsRequest.RegisterStructRequest . . . 130
6.76.1 Подробное описание . . . 131
6.76.2 Данные класса . . . 131
6.76.2.1 login . . . 131
6.76.2.2 password . . . 131
6.77 Класс Room . . . 131
6.77.1 Подробное описание . . . 131
6.77.2 Методы . . . 132
6.77.2.1 RotateRandomly() . . . 132
6.78 Класс Setting . . . 132
6.78.1 Подробное описание . . . 132
6.78.2 Конструктор(ы) . . . 132
6.78.2.1 Setting() . . . 132
6.78.3 Данные класса . . . 133
6.78.3.1 name . . . 133
6.78.3.2 value . . . 133
6.79 Класс SettingsManager . . . 133
6.79.1 Подробное описание . . . 134
6.79.2 Методы . . . 134
6.79.2.1 ChangeFps() . . . 134
6.79.2.2 ChangeFullscreen() . . . 135
6.79.2.3 ChangeResolution() . . . 135
6.79.2.4 ChangeVsync() . . . 135
6.79.2.5 OnKeybindMovementToggle() . . . 136
6.79.2.6 SwitchPanels() . . . 136

6.79.3 Данные класса . . . 136
6.79.3.1 keybindMovementToggled . . . 136
6.80 Структура BaseEntity.Skill . . . 137
6.80.1 Подробное описание . . . 137
6.80.2 Данные класса . . . 137
6.80.2.1 key . . . 137
6.80.2.2 skill . . . 137
6.81 Класс SkillCastTimer . . . 137
6.81.1 Подробное описание . . . 138
6.81.2 Методы . . . 138
6.81.2.1 StartTimer() . . . 138
6.82 Класс SkillPanelManager . . . 138
6.82.1 Подробное описание . . . 139
6.82.2 Данные класса . . . 139
6.82.2.1 trackedPlayer . . . 139
6.83 Класс SocketServerController . . . 139
6.83.1 Подробное описание . . . 139
6.83.2 Методы . . . 140
6.83.2.1 CloseConnection() . . . 140
6.83.2.2 StartSendInviteToSocketServer() . . . 140
6.84 Класс ServersInfo.SocketServerInfo . . . 140
6.84.1 Подробное описание . . . 141
6.84.2 Данные класса . . . 141
6.84.2.1 CommandHaveInvite . . . 141
6.84.2.2 CommandHaveInviteServer . . . 141
6.84.2.3 CommandInvite . . . 141
6.84.2.4 CommandInviteServer . . . 141
6.84.2.5 CommandOpenConnect . . . 141
6.84.2.6 CommandOpenConnectServer . . . 142
6.84.2.7 SocketServerIP . . . 142
6.84.2.8 SocketServerPort . . . 142
6.85 Класс SoundManager . . . 142
6.85.1 Подробное описание . . . 142
6.86 Класс SoundSettings . . . 143
6.86.1 Подробное описание . . . 143
6.86.2 Методы . . . 143
6.86.2.1 GetVolume() . . . 143
6.86.2.2 SoundSliderValueChange() . . . 144
6.86.3 Данные класса . . . 144
6.86.3.1 instance . . . 144
6.86.3.2 volumeChange . . . 144
6.87 Класс SoundSliderManager . . . 145
6.87.1 Подробное описание . . . 145

6.87.2 Методы . . . 145
6.87.2.1 SetName() . . . 145
6.87.2.2 SetValue() . . . 146
6.87.2.3 ValueChanged() . . . 146
6.87.3 Данные класса . . . 146
6.87.3.1 onValueChange . . . 146
6.88 Класс SpawnEnemy . . . 147
6.88.1 Подробное описание . . . 147
6.88.2 Данные класса . . . 147
6.88.2.1 goblinSeekerGameObject2 . . . 147
6.89 Класс SpawnPlayers . . . 147
6.89.1 Подробное описание . . . 148
6.89.2 Методы . . . 148
6.89.2.1 SetPlayerObject() . . . 148
6.89.3 Данные класса . . . 148
6.89.3.1 OnSpawn . . . 148
6.90 Класс SpeedUp . . . 148
6.90.1 Подробное описание . . . 149
6.90.2 Методы . . . 149
6.90.2.1 ApplyEffect() . . . 149
6.90.2.2 SpeedEffect() . . . 149
6.90.3 Данные класса . . . 149
6.90.3.1 additionalSpeed . . . 150
6.91 Класс StartEndZone . . . 150
6.91.1 Подробное описание . . . 150
6.92 Класс StartGameLevel . . . 150
6.92.1 Подробное описание . . . 151
6.92.2 Методы . . . 151
6.92.2.1 OnJoinedRoom() . . . 151
6.92.2.2 OnPlayerEnteredRoom() . . . 151
6.92.2.3 OnPlayerLeftRoom() . . . 151
6.93 Класс StopSearchGame . . . 152
6.93.1 Подробное описание . . . 152
6.93.2 Методы . . . 152
6.93.2.1 CancelSearchGame() . . . 152
6.93.2.2 OnConnectedToMaster() . . . 153
6.93.2.3 OnFriendListUpdate() . . . 153
6.94 Класс Strong . . . 153
6.94.1 Подробное описание . . . 154
6.94.2 Методы . . . 154
6.94.2.1 ApplyEffect() . . . 154
6.94.2.2 StrongEffect() . . . 154
6.94.3 Данные класса . . . 154

6.94.3.1 additionalDamage . . . 154
6.95 Класс TeamHPBarUI . . . 155
6.95.1 Подробное описание . . . 155
6.95.2 Методы . . . 155
6.95.2.1 SetEntity() . . . 155
6.96 Класс TeamHpPanelManager . . . 156
6.96.1 Подробное описание . . . 156
6.97 Класс TimerManager . . . 156
6.97.1 Подробное описание . . . 156
6.97.2 Данные класса . . . 156
6.97.2.1 timerEnd . . . 157
6.98 Класс TimerWithSpritemanager . . . 157
6.98.1 Подробное описание . . . 157
6.98.2 Методы . . . 157
6.98.2.1 SetEntity() . . . 157
6.98.2.2 SetTime() . . . 158
6.99 Класс StructsResponse.TokenStructResponse . . . 158
6.99.1 Подробное описание . . . 158
6.99.2 Данные класса . . . 158
6.99.2.1 token . . . 159
6.100 Класс TooltipText . . . 159
6.100.1 Подробное описание . . . 159
6.100.2 Данные класса . . . 159
6.100.2.1 text . . . 159
6.101 Класс TooltipTextUI . . . 160
6.101.1 Подробное описание . . . 160
6.101.2 Данные класса . . . 160
6.101.2.1 text . . . 160
6.102 Класс Toxicant . . . 160
6.102.1 Подробное описание . . . 161
6.102.2 Методы . . . 161
6.102.2.1 UseSkill() . . . 161
6.103 Класс UIHintManager . . . 161
6.103.1 Подробное описание . . . 162
6.103.2 Перечисления . . . 162
6.103.2.1 ProjectMode . . . 162
6.103.3 Данные класса . . . 162
6.103.3.1 _camera . . . 163
6.103.3.2 arrow . . . 163
6.103.3.3 BGColor . . . 163
6.103.3.4 border . . . 163
6.103.3.5 box . . . 163
6.103.3.6 boxText . . . 163

6.103.3.7 fontSize . . . 164
6.103.3.8 isUI . . . 164
6.103.3.9 maxWidth . . . 164
6.103.3.10 speed . . . 164
6.103.3.11 text . . . 164
6.103.3.12 textColor . . . 164
6.103.3.13 tooltipMode . . . 165
6.104 Класс UnityMainThreadDispatcher . . . 165
6.104.1 Подробное описание . . . 165
6.104.2 Методы . . . 166
6.104.2.1 Enqueue() [1/2] . . . 166
6.104.2.2 Enqueue() [2/2] . . . 166
6.104.2.3 EnqueueAsync() . . . 166
6.104.2.4 Exists() . . . 167
6.104.2.5 Instance() . . . 167
6.104.2.6 Update() . . . 168
6.105 Класс StructsResponse.UserIDStructResponse . . . 168
6.105.1 Подробное описание . . . 168
6.105.2 Данные класса . . . 168
6.105.2.1 userID . . . 168
6.106 Класс StructsResponse.UserInfoObject . . . 168
6.106.1 Подробное описание . . . 169
6.106.2 Данные класса . . . 169
6.106.2.1 userID . . . 169
6.106.2.2 username . . . 169
6.107 Класс StructsResponse.UserInfoStructResponse . . . 169
6.107.1 Подробное описание . . . 169
6.107.2 Данные класса . . . 169
6.107.2.1 created_at . . . 170
6.107.2.2 login . . . 170
6.107.2.3 username . . . 170
6.108 Класс UserOnline . . . 170
6.108.1 Подробное описание . . . 170
6.109 Класс StructsRequest.UserOnlineAndOfflineStructRequest . . . 171
6.109.1 Подробное описание . . . 171
6.109.2 Данные класса . . . 171
6.109.2.1 data . . . 171

7 Файлы 173
7.1 LoginManager.cs . . . 173
7.2 RegisterManager.cs . . . 173
7.3 Controllers.cs . . . 174
7.4 DataBase.cs . . . 177

7.5 AgressionTrigger.cs . . . 180
7.6 AttackTrigger.cs . . . 180
7.7 EnemyAI.cs . . . 180
7.8 EntityMelee.cs . . . 181
7.9 PlayerAOE.cs . . . 182
7.10 PlayerMelee.cs . . . 182
7.11 PlayerProjectile.cs . . . 184
7.12 BaseAOE.cs . . . 185
7.13 BaseEffect.cs . . . 186
7.14 BaseEntity.cs . . . 186
7.15 BaseItem.cs . . . 190
7.16 BasePassiveSkill.cs . . . 191
7.17 BaseProjectile.cs . . . 191
7.18 BaseSkill.cs . . . 193
7.19 BaseUpgrade.cs . . . 194
7.20 CameraFollow.cs . . . 194
7.21 Drunk.cs . . . 194
7.22 OnFire.cs . . . 195
7.23 Poison.cs . . . 195
7.24 SpeedUp.cs . . . 195
7.25 Strong.cs . . . 196
7.26 EndGame.cs . . . 196
7.27 DisplayedItem.cs . . . 196
7.28 Inventory.cs . . . 197
7.29 InventoryWindow.cs . . . 198
7.30 Appe.cs . . . 199
7.31 DroppedItem.cs . . . 199
7.32 ManagerGame.cs . . . 200
7.33 Generator.cs . . . 202
7.34 Room.cs . . . 203
7.35 StartEndZone.cs . . . 203
7.36 Berserk.cs . . . 203
7.37 AnimationPlayer.cs . . . 204
7.38 CameraPlayer.cs . . . 205
7.39 ClassSelection.cs . . . 206
7.40 InputControllerPlayer.cs . . . 207
7.41 MovementPlayer.cs . . . 207
7.42 PlayerScript.cs . . . 209
7.43 PlayerUpgrades.cs . . . 209
7.44 Drunkard.cs . . . 210
7.45 Exclamation.cs . . . 211
7.46 FireballSkill.cs . . . 211
7.47 FlurryOfFire.cs . . . 212

7.48 Toxicant.cs . . . 212
7.49 SoundManager.cs . . . 213
7.50 SpawnEnemy.cs . . . 213
7.51 SpawnPlayers.cs . . . 214
7.52 PlayersTeamsManager.cs . . . 214
7.53 Connect.cs . . . 215
7.54 AcceptFriendInvite.cs . . . 215
7.55 FriendsListController.cs . . . 216
7.56 InviteFriend.cs . . . 217
7.57 LeaveRoom.cs . . . 217
7.58 LobbyManager.cs . . . 217
7.59 NoticeListController.cs . . . 219
7.60 Play.cs . . . 219
7.61 PlayersFounded.cs . . . 221
7.62 SocketServerController.cs . . . 221
7.63 StartGameLevel.cs . . . 223
7.64 StopSearchGame.cs . . . 224
7.65 Structs.cs . . . 225
7.66 DISABLESUKA.cs . . . 226
7.67 ENABLESUKA.cs . . . 227
7.68 KeyHandler.cs . . . 227
7.69 EffectsListManager.cs . . . 229
7.70 EffectTimerManager.cs . . . 230
7.71 EndGameManager.cs . . . 231
7.72 EntityNameManager.cs . . . 231
7.73 EscapeMenu.cs . . . 232
7.74 FriendItemManager.cs . . . 232
7.75 HealthBarManager.cs . . . 233
7.76 KeybindManager.cs . . . 233
7.77 NoticeInviteManager.cs . . . 234
7.78 SettingsManager.cs . . . 235
7.79 SkillPanelManager.cs . . . 237
7.80 SoundSettings.cs . . . 238
7.81 SoundSliderManager.cs . . . 238
7.82 TeamHpPanelManager.cs . . . 239
7.83 TimerManager.cs . . . 239
7.84 TimerWithSpritemanager.cs . . . 240
7.85 UIHintManager.cs . . . 240
7.86 EnergyBarUI.cs . . . 242
7.87 HealthBarUI.cs . . . 243
7.88 SkillCastTimer.cs . . . 243
7.89 TeamHPBarUI.cs . . . 244
7.90 TooltipText.cs . . . 244

7.91 TooltipTextUI.cs 245
7.92 UnityMainThreadDispatcher.cs 245
7.93 UserOnline.cs 246

Предметный указатель 247

Глава 1

Алфавитный указатель пространств имен

1.1 Пространства имен

Полный список документированных пространств имен.

GameSettings 13
ServersInfo 13
StructsRequest 13
StructsResponse 13

Глава 2

Иерархический список классов

2.1 Иерархия классов

Иерархия классов.

PlayerAOE.AOEstruct 20
EventTrigger
  DisplayedItem 59
StructsRequest.FindFriendsByNameStructRequest 76
StructsResponse.FindFriendsByNameStructResponse 77
GameSettings.GameSettingsOriginal 83
IPointerEnterHandler
  TooltipTextUI 160
IPointerExitHandler
  TooltipTextUI 160
Keybind 93
StructsRequest.LoginStructRequest 109
ServersInfo.MainServerInfo 110
MonoBehaviour
  AgressionTrigger 17
  AnimationPlayer 17
  AttackTrigger 22
  BaseAOE 22
  BaseEffect 24
    Drunk 62
    OnFire 115
    Poison 126
    SpeedUp 148
    Strong 153
  BaseEntity 26
  BasePassiveSkill 37
    Berserk 46
  BaseProjectile 39
  BaseSkill 41
    Drunkard 64
    Exclamation 75
    FireballSkill 78
    FlurryOfFire 79
    Toxicant 160
  BaseUpgrade 45

CameraFollow 47
CameraPlayer 47
ClassSelection 48
Controllers 49
DISABLESUKA 58
DataBase 53
DroppedItem 61
ENABLESUKA 67
EffectTimerManager 66
EffectsListManager 65
EndGame 67
EndGameManager 68
EnemyAI 69
EnergyBarUI 70
EntityMelee 71
EntityNameManager 72
EscapeMenu 74
FriendsListController 81
Generator 84
HealthBarManager 85
HealthBarUI 87
InputControllerPlayer 88
Inventory 88
InventoryWindow 91
KeyHandler 98
KeybindManager 94
LoginManager 108
MovementPlayer 111
NoticeInviteManager 112
NoticeListController 114
PlayerAOE 118
PlayerMelee 119
PlayerProjectile 120
PlayerScript 121
PlayerUpgrades 125
PlayersFounded 122
PlayersTeamsManager 124
RegisterManager 129
Room 131
SettingsManager 133
SkillCastTimer 137
SkillPanelManager 138
SocketServerController 139
SoundManager 142
SoundSettings 143
SoundSliderManager 145
SpawnEnemy 147
SpawnPlayers 147
StartEndZone 150
TeamHPBarUI 155
TeamHpPanelManager 156
TimerManager 156
TimerWithSpritemanager 157
TooltipText 159
TooltipTextUI 160
UIHintManager 161
UnityMainThreadDispatcher 165
UserOnline 170

MonoBehaviourPunCallbacks
AcceptFriendInvite . . . 15
Connect . . . 48
FriendItemManager . . . 80
InviteFriend . . . 92
LeaveRoom . . . 104
LobbyManager . . . 105
ManagerGame . . . 110
Play . . . 116
StartGameLevel . . . 150
StopSearchGame . . . 152
PlayerProjectile.Projectile . . . 128
StructsRequest.RegisterStructRequest . . . 130
ScriptableObject
BaseItem . . . 35
Appe . . . 21
Setting . . . 132
BaseEntity.Skill . . . 137
ServersInfo.SocketServerInfo . . . 140
StructsResponse.TokenStructResponse . . . 158
StructsResponse.UserIDStructResponse . . . 168
StructsResponse.UserInfoObject . . . 168
StructsResponse.UserInfoStructResponse . . . 169
StructsRequest.UserOnlineAndOfflineStructRequest . . . 171

Глава 3

Алфавитный указатель классов

3.1 Классы

Классы с их кратким описанием.

AcceptFriendInvite . . . 15
AgressionTrigger . . . 17
AnimationPlayer . . . 17
PlayerAOE.AOEstruct . . . 20
Appe . . . 21
AttackTrigger . . . 22
BaseAOE . . . 22
BaseEffect . . . 24
BaseEntity . . . 26
BaseItem . . . 35
BasePassiveSkill . . . 37
BaseProjectile . . . 39
BaseSkill . . . 41
BaseUpgrade . . . 45
Berserk . . . 46
CameraFollow . . . 47
CameraPlayer . . . 47
ClassSelection
Модуль управления панелью выбора класса пользователя . . . 48
Connect . . . 48
Controllers
Класс для взаимодействий с сервером Vuji . . . 49
DataBase
Модуль взаимодействия с базой данных игры . . . 53
DISABLESUKA
Вспомогательный скрипт, выключает все объекты (список задается в редакторе юнити) . . . 58
DisplayedItem
Модуль для управления отображаемым предметом инвентаря . . . 59
DroppedItem . . . 61
Drunk . . . 62
Drunkard . . . 64
EffectsListManager
Модуль по управлению списком эффектов . . . 65
EffectTimerManager
Класс для управления таймером эффекта . . . 66

ENABLESUKA
Вспомогательный скрипт, включает все объекты (список задается в редакторе юнити) . . . 67
EndGame
Модуль для отслеживания окончания игры . . . 67
EndGameManager
Вспомогательный модуль для отображения панели после окончания игры . . . 68
EnemyAI . . . 69
EnergyBarUI
Модуль управления баром энергии в нижней части экрана . . . 70
EntityMelee . . . 71
EntityNameManager
Модуль для упраления и корректировки позиции имени над сущностью . . . 72
EscapeMenu
Модуль для управления меню паузы . . . 74
Exclamation . . . 75
StructsRequest.FindFriendsByNameStructRequest . . . 76
StructsResponse.FindFriendsByNameStructResponse . . . 77
FireballSkill . . . 78
FlurryOfFire . . . 79
FriendItemManager
Модуль управления объектом пользователя, доступного для приглашения в группу (в лобби) . . . 80
FriendsListController . . . 81
GameSettings.GameSettingsOriginal . . . 83
Generator . . . 84
HealthBarManager
Модуль для управления полоской жизней над сущностью . . . 85
HealthBarUI
Модуль управления баром хп в нижней части экрана . . . 87
InputControllerPlayer . . . 88
Inventory . . . 88
InventoryWindow
Модуль управления отображаемым инвентарем сущности . . . 91
InviteFriend . . . 92
Keybind
Вспомогательный класс для передачи информации о сохраненных настройках управления . . . 93
KeybindManager
Модуль управления объектом настройки управления . . . 94
KeyHandler
Модуль для всего, что связанно со считыванием нажатий на клавиатуру (привязка, считывание, регулирование) . . . 98
LeaveRoom . . . 104
LobbyManager . . . 105
LoginManager . . . 108
StructsRequest.LoginStructRequest . . . 109
ServersInfo.MainServerInfo . . . 110
ManagerGame . . . 110
MovementPlayer . . . 111
NoticeInviteManager
Модуль управления списком приглашений в группу (в лобби) . . . 112
NoticeListController . . . 114
OnFire . . . 115
Play . . . 116
PlayerAOE . . . 118
PlayerMelee . . . 119
PlayerProjectile . . . 120

PlayerScript . . . 121
PlayersFounded . . . 122
PlayersTeamsManager . . . 124
PlayerUpgrades
Модуль управления панелью улучшения игрока !НЕ АКТИВНО! . . . 125
Poison . . . 126
PlayerProjectile.Projectile . . . 128
RegisterManager . . . 129
StructsRequest.RegisterStructRequest . . . 130
Room . . . 131
Setting
Вспомогательный класс для передачи сохраненных настроек . . . 132
SettingsManager
Модуль управления настройками . . . 133
BaseEntity.Skill . . . 137
SkillCastTimer
Модуль для управления таймером каста скила целевой сущностью . . . 137
SkillPanelManager
Модуль для управления панелью скилов сущности . . . 138
SocketServerController . . . 139
ServersInfo.SocketServerInfo . . . 140
SoundManager
Модуль для уравления звуком целевого источника . . . 142
SoundSettings
Модуль для управления настройками звука . . . 143
SoundSliderManager
Модуль управления слайдером для настройки звука . . . 145
SpawnEnemy . . . 147
SpawnPlayers . . . 147
SpeedUp . . . 148
StartEndZone . . . 150
StartGameLevel . . . 150
StopSearchGame . . . 152
Strong . . . 153
TeamHPBarUI
Модуль для управления отдельным хп баром в панели списка участников команды 155
TeamHpPanelManager
Модуль управления панелью списка участников команды . . . 156
TimerManager
Модуль управления таймером выбора класса . . . 156
TimerWithSpritemanager
Модуль для управления отображаемым объектом скила . . . 157
StructsResponse.TokenStructResponse . . . 158
TooltipText
Модуль для игровых объектов, при наведении на которых будет показана подсказка . . . 159
TooltipTextUI
Модуль для объектов UI, при наведении на которых будет показана подсказка . . . 160
Toxicant . . . 160
UIHintManager
Вспомогательный модуль для отображения подсказок на экране. Для отображения подсказки на объекте добавьте скрипт TooltipText (Для Ui - TooltipTextUI) . . . 161
UnityMainThreadDispatcher
A thread-safe class which holds a queue with actions to execute on the next Update() method. It can be used to make calls to the main thread for things such as UI Manipulation in Unity. It was developed for use in combination with the Firebase Unity plugin, which uses separate threads for event handling . . . 165
StructsResponse.UserIDStructResponse . . . 168

StructsResponse.UserInfoObject 168
StructsResponse.UserInfoStructResponse 169
UserOnline
Вспомогательный модуль для сообщения серверу о том, что пользователь онлайн 170
StructsRequest.UserOnlineAndOfflineStructRequest 171

Глава 4

Список файлов

4.1 Файлы

Полный список документированных файлов.

Vuji/Assets/Scripts/Controllers.cs . . . 174
Vuji/Assets/Scripts/DataBase.cs . . . 177
Vuji/Assets/Scripts/Structs.cs . . . 225
Vuji/Assets/Scripts/UnityMainThreadDispatcher.cs . . . 245
Vuji/Assets/Scripts/UserOnline.cs . . . 246
Vuji/Assets/Scripts/Authorization/Login/LoginManager.cs . . . 173
Vuji/Assets/Scripts/Authorization/Register/RegisterManager.cs . . . 173
Vuji/Assets/Scripts/Game/EndGame.cs . . . 196
Vuji/Assets/Scripts/Game/ManagerGame.cs . . . 200
Vuji/Assets/Scripts/Game/AI/AgressionTrigger.cs . . . 180
Vuji/Assets/Scripts/Game/AI/AttackTrigger.cs . . . 180
Vuji/Assets/Scripts/Game/AI/EnemyAI.cs . . . 180
Vuji/Assets/Scripts/Game/AI/EntityMelee.cs . . . 181
Vuji/Assets/Scripts/Game/Attack/PlayerAOE.cs . . . 182
Vuji/Assets/Scripts/Game/Attack/PlayerMelee.cs . . . 182
Vuji/Assets/Scripts/Game/Attack/PlayerProjectile.cs . . . 184
Vuji/Assets/Scripts/Game/BaseObjects/BaseAOE.cs . . . 185
Vuji/Assets/Scripts/Game/BaseObjects/BaseEffect.cs . . . 186
Vuji/Assets/Scripts/Game/BaseObjects/BaseEntity.cs . . . 186
Vuji/Assets/Scripts/Game/BaseObjects/BaseItem.cs . . . 190
Vuji/Assets/Scripts/Game/BaseObjects/BasePassiveSkill.cs . . . 191
Vuji/Assets/Scripts/Game/BaseObjects/BaseProjectile.cs . . . 191
Vuji/Assets/Scripts/Game/BaseObjects/BaseSkill.cs . . . 193
Vuji/Assets/Scripts/Game/BaseObjects/BaseUpgrade.cs . . . 194
Vuji/Assets/Scripts/Game/Camera/CameraFollow.cs . . . 194
Vuji/Assets/Scripts/Game/Effects/Drunk.cs . . . 194
Vuji/Assets/Scripts/Game/Effects/OnFire.cs . . . 195
Vuji/Assets/Scripts/Game/Effects/Poison.cs . . . 195
Vuji/Assets/Scripts/Game/Effects/SpeedUp.cs . . . 195
Vuji/Assets/Scripts/Game/Effects/Strong.cs . . . 196
Vuji/Assets/Scripts/Game/Inventory/DisplayedItem.cs . . . 196
Vuji/Assets/Scripts/Game/Inventory/Inventory.cs . . . 197
Vuji/Assets/Scripts/Game/Inventory/InventoryWindow.cs . . . 198
Vuji/Assets/Scripts/Game/Items/Appe.cs . . . 199
Vuji/Assets/Scripts/Game/Items/DroppedItem.cs . . . 199

Vuji/Assets/Scripts/Game/Map/Generator.cs . . . 202
Vuji/Assets/Scripts/Game/Map/Room.cs . . . 203
Vuji/Assets/Scripts/Game/Map/StartEndZone.cs . . . 203
Vuji/Assets/Scripts/Game/PassiveSkills/Berserk.cs . . . 203
Vuji/Assets/Scripts/Game/Player/AnimationPlayer.cs . . . 204
Vuji/Assets/Scripts/Game/Player/CameraPlayer.cs . . . 205
Vuji/Assets/Scripts/Game/Player/ClassSelection.cs . . . 206
Vuji/Assets/Scripts/Game/Player/InputControllerPlayer.cs . . . 207
Vuji/Assets/Scripts/Game/Player/MovementPlayer.cs . . . 207
Vuji/Assets/Scripts/Game/Player/PlayerScript.cs . . . 209
Vuji/Assets/Scripts/Game/Player/PlayerUpgrades.cs . . . 209
Vuji/Assets/Scripts/Game/Skills/Drunkard.cs . . . 210
Vuji/Assets/Scripts/Game/Skills/Exclamation.cs . . . 211
Vuji/Assets/Scripts/Game/Skills/FireballSkill.cs . . . 211
Vuji/Assets/Scripts/Game/Skills/FlurryOfFire.cs . . . 212
Vuji/Assets/Scripts/Game/Skills/Toxicant.cs . . . 212
Vuji/Assets/Scripts/Game/Sounds/SoundManager.cs . . . 213
Vuji/Assets/Scripts/Game/Spawn/SpawnEnemy.cs . . . 213
Vuji/Assets/Scripts/Game/Spawn/SpawnPlayers.cs . . . 214
Vuji/Assets/Scripts/Game/Teams/PlayersTeamsManager.cs . . . 214
Vuji/Assets/Scripts/Loading/Connect.cs . . . 215
Vuji/Assets/Scripts/Lobby/AcceptFriendInvite.cs . . . 215
Vuji/Assets/Scripts/Lobby/FriendsListController.cs . . . 216
Vuji/Assets/Scripts/Lobby/InviteFriend.cs . . . 217
Vuji/Assets/Scripts/Lobby/LeaveRoom.cs . . . 217
Vuji/Assets/Scripts/Lobby/LobbyManager.cs . . . 217
Vuji/Assets/Scripts/Lobby/NoticeListController.cs . . . 219
Vuji/Assets/Scripts/Lobby/Play.cs . . . 219
Vuji/Assets/Scripts/Lobby/PlayersFounded.cs . . . 221
Vuji/Assets/Scripts/Lobby/SocketServerController.cs . . . 221
Vuji/Assets/Scripts/Lobby/StartGameLevel.cs . . . 223
Vuji/Assets/Scripts/Lobby/StopSearchGame.cs . . . 224
Vuji/Assets/Scripts/UIScripts/DISABLESUKA.cs . . . 226
Vuji/Assets/Scripts/UIScripts/ENABLESUKA.cs . . . 227
Vuji/Assets/Scripts/UIScripts/KeyHandler.cs . . . 227
Vuji/Assets/Scripts/UIScripts/Managers/EffectsListManager.cs . . . 229
Vuji/Assets/Scripts/UIScripts/Managers/EffectTimerManager.cs . . . 230
Vuji/Assets/Scripts/UIScripts/Managers/EndGameManager.cs . . . 231
Vuji/Assets/Scripts/UIScripts/Managers/EntityNameManager.cs . . . 231
Vuji/Assets/Scripts/UIScripts/Managers/EscapeMenu.cs . . . 232
Vuji/Assets/Scripts/UIScripts/Managers/FriendItemManager.cs . . . 232
Vuji/Assets/Scripts/UIScripts/Managers/HealthBarManager.cs . . . 233
Vuji/Assets/Scripts/UIScripts/Managers/KeybindManager.cs . . . 233
Vuji/Assets/Scripts/UIScripts/Managers/NoticeInviteManager.cs . . . 234
Vuji/Assets/Scripts/UIScripts/Managers/SettingsManager.cs . . . 235
Vuji/Assets/Scripts/UIScripts/Managers/SkillPanelManager.cs . . . 237
Vuji/Assets/Scripts/UIScripts/Managers/SoundSettings.cs . . . 238
Vuji/Assets/Scripts/UIScripts/Managers/SoundSliderManager.cs . . . 238
Vuji/Assets/Scripts/UIScripts/Managers/TeamHpPanelManager.cs . . . 239
Vuji/Assets/Scripts/UIScripts/Managers/TimerManager.cs . . . 239
Vuji/Assets/Scripts/UIScripts/Managers/TimerWithSpritemanager.cs . . . 240
Vuji/Assets/Scripts/UIScripts/Managers/UIHintManager.cs . . . 240
Vuji/Assets/Scripts/UIScripts/Units/EnergyBarUI.cs . . . 242
Vuji/Assets/Scripts/UIScripts/Units/HealthBarUI.cs . . . 243
Vuji/Assets/Scripts/UIScripts/Units/SkillCastTimer.cs . . . 243
Vuji/Assets/Scripts/UIScripts/Units/TeamHPBarUI.cs . . . 244
Vuji/Assets/Scripts/UIScripts/Units/TooltipText.cs . . . 244
Vuji/Assets/Scripts/UIScripts/Units/TooltipTextUI.cs . . . 245

Глава 5

Пространства имен

5.1 Пространство имен GameSettings

Классы

- class GameSettingsOriginal

5.2 Пространство имен ServersInfo

Классы

- class MainServerInfo
- class SocketServerInfo

5.3 Пространство имен StructsRequest

Классы

- class FindFriendsByNameStructRequest
- class LoginStructRequest
- class RegisterStructRequest
- class UserOnlineAndOfflineStructRequest

5.4 Пространство имен StructsResponse

Классы

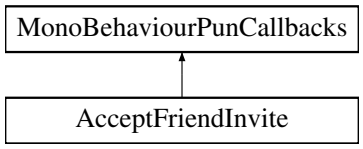
- class FindFriendsByNameStructResponse
- class TokenStructResponse
- class UserIDStructResponse
- class UserInfoObject
- class UserInfoStructResponse

Глава 6

Классы

6.1 Класс AcceptFriendInvite

Граф наследования:AcceptFriendInvite:



Открытые члены

- void StartAcceptInvite ()

  Метод на подключение к другому лобби
- override void OnConnectedToMaster ()
- override void OnJoinedRoom ()

Открытые атрибуты

- string roomName

6.1.1 Подробное описание

См. определение в файле AcceptFriendInvite.cs строка 3

6.1.2 Методы

6.1.2.1 OnConnectedToMaster()

override void AcceptFriendInvite.OnConnectedToMaster ( ) [inline]

См. определение в файле AcceptFriendInvite.cs строка 23

```
00024     {
00025         StartAcceptInvite();
00026     }
```

6.1.2.2 OnJoinedRoom()

override void AcceptFriendInvite.OnJoinedRoom ( ) [inline]

См. определение в файле AcceptFriendInvite.cs строка 28

```
00029     {
00030         enabled = false;
00031     }
```

6.1.2.3 StartAcceptInvite()

void AcceptFriendInvite.StartAcceptInvite ( ) [inline]

Метод на подключение к другому лобби

См. определение в файле AcceptFriendInvite.cs строка 10

```
00011     {
00012         if (PhotonNetwork.InRoom)
00013         {
00014             PhotonNetwork.LeaveRoom();
00015         }
00016         else
00017         {
00018             gameObject.GetComponent<LobbyManager>().playerStatus = "INLOBBY";
00019             PhotonNetwork.JoinRoom(roomName);
00020         }
00021     }
```

6.1.3 Данные класса

6.1.3.1 roomName

string AcceptFriendInvite.roomName

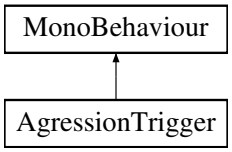
См. определение в файле AcceptFriendInvite.cs строка 5

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Lobby/AcceptFriendInvite.cs

6.2 Класс AgressionTrigger

Граф наследования:AgressionTrigger:



6.2.1 Подробное описание

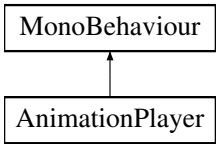
См. определение в файле AgressionTrigger.cs строка 5

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/AI/AgressionTrigger.cs

6.3 Класс AnimationPlayer

Граф наследования:AnimationPlayer:



Открытые члены

- void ChangePlayerAnimation_q (string newAnim)
- string getCurrentMovingState ()
- void ChangePlayerAnimation (string newAnimation)

Открытые атрибуты

- string movingState
- readonly string _attack = "attack_"
- readonly string _move = "move_"
- readonly string _drink = "drink_"
- readonly string _idle = "idle_"
- readonly string _shot = "shot_"
- readonly string _left = "left"
- readonly string _right = "right"
- readonly string _up = "back"
- readonly string _down = "front"

6.3.1 Подробное описание

См. определение в файле AnimationPlayer.cs строка 7

6.3.2 Методы

6.3.2.1 ChangePlayerAnimation()

```
void AnimationPlayer.ChangePlayerAnimation (
                string newAnimation )   [inline]
```

См. определение в файле AnimationPlayer.cs строка 105

```
00106     {
00107         if (_currentAnimation == newAnimation) return;
00108         _currentAnimation = newAnimation;
00109         //Debug.Log(newAnimation+"Photon");
00110         _animator.Play(_currentAnimation);
00111     }
```

6.3.2.2 ChangePlayerAnimation_q()

```
void AnimationPlayer.ChangePlayerAnimation_q (
                string newAnim )   [inline]
```

См. определение в файле AnimationPlayer.cs строка 63

```
00064     {
00065         if(!_animator.GetCurrentAnimatorStateInfo(-1).IsName(_attack+movingState))
00066             _view.RPC("ChangePlayerAnimation", RpcTarget.All, newAnim + movingState);
00067     }
```

6.3.2.3 getCurrentMovingState()

```
string AnimationPlayer.getCurrentMovingState ( )   [inline]
```

См. определение в файле AnimationPlayer.cs строка 68

```
00069     {
00070         y = Input.GetAxisRaw("Vertical");
00071         x = Input.GetAxisRaw("Horizontal");
00072 
00073         if (x != 0 || y != 0)
00074         {
00075             isMoving = true;
00076             if (y > 0)
00077                 movingState = _up;
00078             else
00079                 movingState = _down;
00080             if (x > 0)
00081                 movingState = _right;
00082             else if (x != 0)
00083                 movingState = _left;
00084         }
00085         return movingState;
00086     }
```

6.3.3 Данные класса

6.3.3.1 _attack

readonly string AnimationPlayer._attack = "attack_"

См. определение в файле AnimationPlayer.cs строка 21

6.3.3.2 _down

readonly string AnimationPlayer._down = "front"

См. определение в файле AnimationPlayer.cs строка 30

6.3.3.3 _drink

readonly string AnimationPlayer._drink = "drink_"

См. определение в файле AnimationPlayer.cs строка 23

6.3.3.4 _idle

readonly string AnimationPlayer._idle = "idle_"

См. определение в файле AnimationPlayer.cs строка 24

6.3.3.5 _left

readonly string AnimationPlayer._left = "left"

См. определение в файле AnimationPlayer.cs строка 27

6.3.3.6 _move

readonly string AnimationPlayer._move = "move_"

См. определение в файле AnimationPlayer.cs строка 22

6.3.3.7 _right

readonly string AnimationPlayer._right = "right"

См. определение в файле AnimationPlayer.cs строка 28

6.3.3.8 _shot

readonly string AnimationPlayer._shot = "shot_"

См. определение в файле AnimationPlayer.cs строка 25

6.3.3.9 _up

readonly string AnimationPlayer._up = "back"

См. определение в файле AnimationPlayer.cs строка 29

6.3.3.10 movingState

string AnimationPlayer.movingState

См. определение в файле AnimationPlayer.cs строка 19

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Player/AnimationPlayer.cs

6.4 Структура PlayerAOE.AOEstruct

Открытые атрибуты

- string AOEKey
- GameObject AOE

6.4.1 Подробное описание

См. определение в файле PlayerAOE.cs строка 13

6.4.2 Данные класса

6.4.2.1 AOE

GameObject PlayerAOE.AOEstruct.AOE

См. определение в файле PlayerAOE.cs строка 16

6.4.2.2 AOEKey

string PlayerAOE.AOEstruct.AOEKey

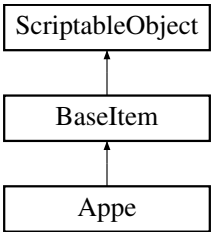
См. определение в файле PlayerAOE.cs строка 15

Объявления и описания членов структуры находятся в файле:

- Vuji/Assets/Scripts/Game/Attack/PlayerAOE.cs

6.5 Класс Appe

Граф наследования:Appe:



Открытые члены

- override bool UseItem (GameObject owner)

6.5.1 Подробное описание

См. определение в файле Appe.cs строка 8

6.5.2 Методы

6.5.2.1 UseItem()

```
override bool Appe.UseItem (
                GameObject owner )   [inline], [virtual]
```

Переопределяет метод предка BaseItem.

См. определение в файле Appe.cs строка 10

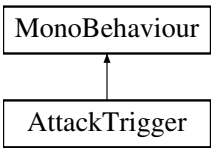
```
00011     {
00012         var _view = owner.GetComponent<PhotonView>();
00013         owner.GetComponent<BaseEntity>().GetHealthPoints();
00014         _view.RPC("Heal", RpcTarget.All, 100.0f);
00015         return base.UseItem(owner);
00016     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Items/Appe.cs

6.6 Класс AttackTrigger

Граф наследования:AttackTrigger:



6.6.1 Подробное описание

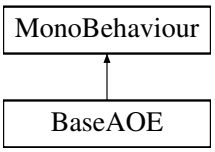
См. определение в файле AttackTrigger.cs строка 5

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/AI/AttackTrigger.cs

6.7 Класс BaseAOE

Граф наследования:BaseAOE:



Открытые члены

- void SetSenderCollider (GameObject senderObject)
- string GetAoeName ()
- string GetAoeDescription ()
- void SetDestroyTags (List< string > newDestroyTags)
- void SetDamageTags (List< string > newDamageTags)

6.7.1 Подробное описание

См. определение в файле BaseAOE.cs строка 7

6.7.2 Методы

6.7.2.1 GetAoeDescription()

string BaseAOE.GetAoeDescription ( ) [inline]

См. определение в файле BaseAOE.cs строка 35

```
00036     {
00037         return aoeDescription;
00038     }
```

6.7.2.2 GetAoeName()

string BaseAOE.GetAoeName ( ) [inline]

См. определение в файле BaseAOE.cs строка 30

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

6.7.2.3 SetDamageTags()

void BaseAOE.SetDamageTags (
List< string > newDamageTags) [inline]

См. определение в файле BaseAOE.cs строка 45

```
00046     {
00047         _damageTags = newDamageTags;
00048     }
```

6.7.2.4 SetDestroyTags()

```
void BaseAOE.SetDestroyTags (
                List< string > newDestroyTags )  [inline]
```

См. определение в файле BaseAOE.cs строка 40

```
00041     {
00042         _destroyTags = newDestroyTags;
00043     }
```

6.7.2.5 SetSenderCollider()

```
void BaseAOE.SetSenderCollider (
                GameObject senderObject )  [inline]
```

См. определение в файле BaseAOE.cs строка 25

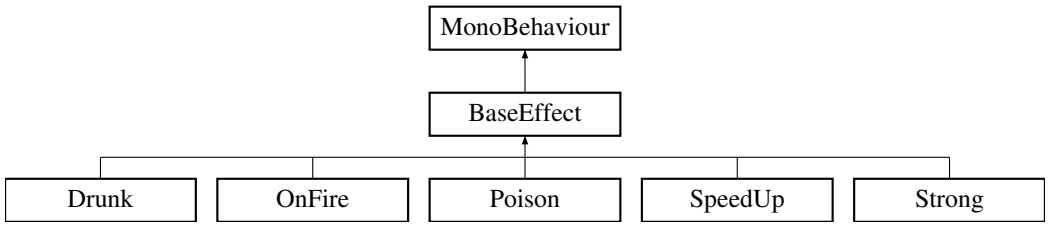
```
00026     {
00027         senderGameObject = senderObject;
00028     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/BaseObjects/BaseAOE.cs

6.8 Класс BaseEffect

Граф наследования:BaseEffect:



Открытые члены

- virtual void ApplyEffect (GameObject entity)

Открытые атрибуты

- string effectName = "New Effect"
- string description = "Effect description"
- float duration = 5f
- Sprite effectSprite

6.8.1 Подробное описание

См. определение в файле BaseEffect.cs строка 5

6.8.2 Методы

6.8.2.1 ApplyEffect()

virtual void BaseEffect.ApplyEffect (
GameObject entity) [inline], [virtual]

См. определение в файле BaseEffect.cs строка 13

```
00014     {
00015         Debug.Log("Apply Effect " + effectName + " on entity " + entity.name);
00016     }
```

6.8.3 Данные класса

6.8.3.1 description

string BaseEffect.description = "Effect description"

См. определение в файле BaseEffect.cs строка 8

6.8.3.2 duration

float BaseEffect.duration = 5f

См. определение в файле BaseEffect.cs строка 9

6.8.3.3 effectName

string BaseEffect.effectName = "New Effect"

См. определение в файле BaseEffect.cs строка 7

6.8.3.4 effectSprite

Sprite BaseEffect.effectSprite

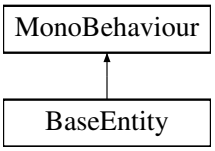
См. определение в файле BaseEffect.cs строка 11

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/BaseObjects/BaseEffect.cs

6.9 Класс BaseEntity

Граф наследования:BaseEntity:



Классы

- struct Skill

Открытые члены

- void UpdateText (string teamTag, string newText)
- void IncreaseMaxHealth (float toAdd)
- void IncreaseMaxEnergy (float toAdd)
- void Heal (float hp)
- void AddEffect (GameObject effect)
- void TickPoints ()
- void UseSkill ()
- void selectSkill (string skillName)
- void deSelectSkill ()
- string GetSelectedSkill ()
- void setIsCooldown (string key, bool value)
- float GetMoveSpeed ()
- float GetHealthPoints ()
- float GetMaxHealthPoints ()
- float GetEnergyPoints ()
- float GetMaxEnergyPoints ()
- int GetBaseDamage ()
- string GetEntityName ()
- int GetDefense ()
- void IncreaseDamage (int addDamage)
- void DecreaseDamage (int addDamage)
- void IncreaseSpeed (int addSpeed)
- void DecreaseSpeed (int addSpeed)
- void IncreaseDefense (int addDefense)
- void DecreaseDefense (int addDefense)
- bool spendEnergy (float energyCost)
- void TakeDamage (int healthDamage)

Открытые атрибуты

- Skill[] skills
- Action< string, bool > OnSkillSelectionChange
- GameObject _droppedItemPrefab
- HealthBarManager healthBar
- EntityNameManager displayedName
- Controllers _controller
- Action< BaseEffect, BaseEntity > OnEffectApply

Статические открытые данные

- static Action< BaseEntity, string > teamSpawn

Свойства

- bool isDead = false [get]

6.9.1 Подробное описание

См. определение в файле BaseEntity.cs строка 9

6.9.2 Методы

6.9.2.1 AddEffect()

```
void BaseEntity.AddEffect (
                GameObject effect )  [inline]
```

См. определение в файле BaseEntity.cs строка 128

```
00129     {
00130         OnEffectApply?.Invoke(effect.GetComponent<BaseEffect>(), this);
00131         effect.GetComponent<BaseEffect>().ApplyEffect(this.gameObject);
00132     }
```

6.9.2.2 DecreaseDamage()

```
void BaseEntity.DecreaseDamage (
                int addDamage )  [inline]
```

См. определение в файле BaseEntity.cs строка 237

```
00238     {
00239         this.baseDamage -= addDamage;
00240     }
```

6.9.2.3 DecreaseDefense()

void BaseEntity.DecreaseDefense (
int addDefense) [inline]

См. определение в файле BaseEntity.cs строка 257

```
00258     {
00259         this.defense -= addDefense;
00260     }
```

6.9.2.4 DecreaseSpeed()

void BaseEntity.DecreaseSpeed (
int addSpeed) [inline]

См. определение в файле BaseEntity.cs строка 247

```
00248     {
00249         this.moveSpeed -= addSpeed;
00250     }
```

6.9.2.5 deSelectSkill()

void BaseEntity.deSelectSkill ( ) [inline]

См. определение в файле BaseEntity.cs строка 172

```
00173     {
00174         Debug.Log("Deselected" + _selectedSkill);
00175         OnSkillSelectionChange?.Invoke(_selectedSkill, false);
00176         this._selectedSkill = "";
00177     }
```

6.9.2.6 GetBaseDamage()

int BaseEntity.GetBaseDamage ( ) [inline]

См. определение в файле BaseEntity.cs строка 216

```
00217     {
00218         return baseDamage;
00219     }
```

6.9.2.7 GetDefense()

int BaseEntity.GetDefense ( ) [inline]

См. определение в файле BaseEntity.cs строка 226

```
00227     {
00228         return defense;
00229     }
```

6.9.2.8 GetEnergyPoints()

float BaseEntity.GetEnergyPoints ( ) [inline]

См. определение в файле BaseEntity.cs строка 206

```
00207     {
00208         return energy;
00209     }
```

6.9.2.9 GetEntityName()

string BaseEntity.GetEntityName ( ) [inline]

См. определение в файле BaseEntity.cs строка 221

```
00222     {
00223         return entityName;
00224     }
```

6.9.2.10 GetHealthPoints()

float BaseEntity.GetHealthPoints ( ) [inline]

См. определение в файле BaseEntity.cs строка 196

```
00197     {
00198         return healthPoints;
00199     }
```

6.9.2.11 GetMaxEnergyPoints()

float BaseEntity.GetMaxEnergyPoints ( ) [inline]

См. определение в файле BaseEntity.cs строка 211

```
00212     {
00213         return maxEnergy;
00214     }
```

6.9.2.12 GetMaxHealthPoints()

float BaseEntity.GetMaxHealthPoints ( ) [inline]

См. определение в файле BaseEntity.cs строка 201

```
00202     {
00203         return maxHealthPoints;
00204     }
```

6.9.2.13 GetMoveSpeed()

float BaseEntity.GetMoveSpeed ( ) [inline]

См. определение в файле BaseEntity.cs строка 191

```
00192     {
00193         return moveSpeed;
00194     }
```

6.9.2.14 GetSelectedSkill()

string BaseEntity.GetSelectedSkill ( ) [inline]

См. определение в файле BaseEntity.cs строка 179

```
00180     {
00181         return _selectedSkill;
00182     }
```

6.9.2.15 Heal()

void BaseEntity.Heal (
float hp) [inline]

См. определение в файле BaseEntity.cs строка 116

```
00116                           {
00117         healthPoints = healthPoints + hp > maxHealthPoints ? maxHealthPoints : healthPoints + hp;
00118     }
```

6.9.2.16 IncreaseDamage()

void BaseEntity.IncreaseDamage (
int addDamage) [inline]

См. определение в файле BaseEntity.cs строка 232

```
00233     {
00234         this.baseDamage += addDamage;
00235     }
```

6.9.2.17 IncreaseDefense()

void BaseEntity.IncreaseDefense (
int addDefense) [inline]

См. определение в файле BaseEntity.cs строка 252

```
00253     {
00254         this.defense += addDefense;
00255     }
```

6.9.2.18 IncreaseMaxEnergy()

void BaseEntity.IncreaseMaxEnergy (
float toAdd) [inline]

См. определение в файле BaseEntity.cs строка 111

```
00112     {
00113         maxEnergy += toAdd;
00114     }
```

6.9.2.19 IncreaseMaxHealth()

void BaseEntity.IncreaseMaxHealth (
float toAdd) [inline]

См. определение в файле BaseEntity.cs строка 106

```
00107     {
00108         maxHealthPoints += toAdd;
00109     }
```

6.9.2.20 IncreaseSpeed()

void BaseEntity.IncreaseSpeed (
int addSpeed) [inline]

См. определение в файле BaseEntity.cs строка 242

```
00243     {
00244         this.moveSpeed += addSpeed;
00245     }
```

6.9.2.21 selectSkill()

void BaseEntity.selectSkill (
string skillName) [inline]

См. определение в файле BaseEntity.cs строка 158

```
00159     {
00160         if ((skillName == "Skill 1" && !_isSkill1Cooldown) || (skillName == "Skill 2" && !_isSkill2Cooldown))
00161         {
00162             Debug.Log("Selected " + skillName);
00163             this._selectedSkill = skillName;
00164             OnSkillSelectionChange?.Invoke(_selectedSkill, true);
00165         }
00166         else
00167         {
00168             Debug.Log(skillName + " COOLDOWN");
00169         }
00170     }
```

6.9.2.22 setIsCooldown()

void BaseEntity.setIsCooldown (
string key,
bool value) [inline]

См. определение в файле BaseEntity.cs строка 184

```
00185     {
00186         if (key == "Skill 1") this._isSkill1Cooldown = value;
00187         if (key == "Skill 2") this._isSkill2Cooldown = value;
00188     }
```

6.9.2.23 spendEnergy()

bool BaseEntity.spendEnergy (
float energyCost) [inline]

См. определение в файле BaseEntity.cs строка 262

```
00263     {
00264         if (energyCost > this.energy) return false;
00265 
00266         this.energy -= energyCost;
00267         return true;
00268     }
```

6.9.2.24 TakeDamage()

void BaseEntity.TakeDamage (
int healthDamage) [inline]

См. определение в файле BaseEntity.cs строка 270

```
00271     {
00272         // Проверка на полное поглощение урона
00273         if(defense - healthDamage >= 0) return;
00274 
00275         healthPoints = healthPoints - healthDamage + defense;
00276         if (healthPoints <= 0)
00277         {
00278             Death();
00279         }
00280 
00281         Debug.Log(entityName + " hp is " + healthPoints);
00282     }
```

6.9.2.25 TickPoints()

void BaseEntity.TickPoints ( ) [inline]

См. определение в файле BaseEntity.cs строка 134

```
00135     {
00136         _currentTick -= Time.deltaTime;
00137         if (_currentTick <= 0)
00138         {
00139             _view.RPC("IncreasePoints", RpcTarget.All);
00140             _currentTick = _regenerationTick;
00141         }
00142     }
```

6.9.2.26 UpdateText()

```
void BaseEntity.UpdateText (
                string teamTag,
                string newText )   [inline]
```

См. определение в файле BaseEntity.cs строка 99

```
00100     {
00101         GetComponentInChildren<Text>().text =  teamTag + newText;
00102         if ("[" + PhotonNetwork.LocalPlayer.GetPhotonTeam().Name + "] " == teamTag) teamSpawn?.Invoke(this,
          newText);
00103     }
```

6.9.2.27 UseSkill()

```
void BaseEntity.UseSkill ( )   [inline]
```

См. определение в файле BaseEntity.cs строка 144

```
00145     {
00146         if (_selectedSkill.StartsWith("Skill"))
00147         {
00148             Debug.Log("Used " + _selectedSkill);
00149             StartCoroutine(_skills[_selectedSkill].GetComponent<BaseSkill>().UseSkill(this.gameObject, _selectedSkill));
00150             deSelectSkill();
00151         }
00152         else
00153         {
00154             Debug.Log("Skill is not selected");
00155         }
00156     }
```

6.9.3 Данные класса

6.9.3.1 _controller

```
Controllers BaseEntity._controller
```

См. определение в файле BaseEntity.cs строка 56

6.9.3.2 _droppedItemPrefab

```
GameObject BaseEntity._droppedItemPrefab
```

См. определение в файле BaseEntity.cs строка 41

6.9.3.3 displayedName

EntityNameManager BaseEntity.displayedName

См. определение в файле BaseEntity.cs строка 54

6.9.3.4 healthBar

HealthBarManager BaseEntity.healthBar

См. определение в файле BaseEntity.cs строка 53

6.9.3.5 OnEffectApply

Action<BaseEffect, BaseEntity> BaseEntity.OnEffectApply

См. определение в файле BaseEntity.cs строка 60

6.9.3.6 OnSkillSelectionChange

Action<string, bool> BaseEntity.OnSkillSelectionChange

См. определение в файле BaseEntity.cs строка 35

6.9.3.7 skills

Skill [] BaseEntity.skills

См. определение в файле BaseEntity.cs строка 32

6.9.3.8 teamSpawn

Action<BaseEntity, string> BaseEntity.teamSpawn [static]

См. определение в файле BaseEntity.cs строка 55

6.9.4 Полный список свойств

6.9.4.1 isDead

bool BaseEntity.isDead = false [get]

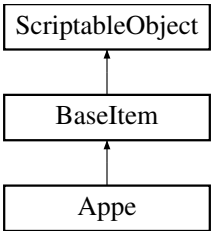
См. определение в файле BaseEntity.cs строка 58
```
00058 { get; private set; } = false;
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/BaseObjects/BaseEntity.cs

6.10 Класс BaseItem

Граф наследования:BaseItem:



Открытые члены

- void SetAmount (int newAmount)
- string GetItemName ()
- string GetDescription ()
- int GetAmount ()
- Sprite GetImage ()
- virtual bool UseItem (GameObject owner)

6.10.1 Подробное описание

См. определение в файле BaseItem.cs строка 6

6.10.2 Методы

6.10.2.1 GetAmount()

int BaseItem.GetAmount ( ) [inline]

См. определение в файле BaseItem.cs строка 33
```
00034     {
00035         return amount;
00036     }
```

6.10.2.2 GetDescription()

string BaseItem.GetDescription ( ) [inline]

См. определение в файле BaseItem.cs строка 28

```
00029     {
00030         return description;
00031     }
```

6.10.2.3 GetImage()

Sprite BaseItem.GetImage ( ) [inline]

См. определение в файле BaseItem.cs строка 38

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

6.10.2.4 GetItemName()

string BaseItem.GetItemName ( ) [inline]

См. определение в файле BaseItem.cs строка 23

```
00024     {
00025         return itemName;
00026     }
```

6.10.2.5 SetAmount()

void BaseItem.SetAmount (
int newAmount) [inline]

См. определение в файле BaseItem.cs строка 13

```
00014     {
00015         if (newAmount < 0)
00016         {
00017             return;
00018         }
00019
00020         amount = newAmount;
00021     }
```

6.10.2.6 UseItem()

```
virtual bool BaseItem.UseItem (
                GameObject owner )   [inline], [virtual]
```

См. определение в файле BaseItem.cs строка 42

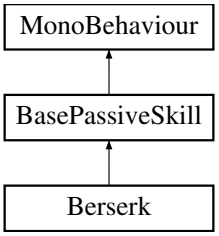
```
00043     {
00044         // Do smth
00045         amount = amount - 1;
00046         if (amount < 1)
00047         {
00048             return false;
00049         }
00050         return true;
00051     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/BaseObjects/BaseItem.cs

6.11 Класс BasePassiveSkill

Граф наследования:BasePassiveSkill:



Открытые члены

- string GetName ()
- string GetDescription ()
- virtual void ActivatePassiveSkill (GameObject caster)

Защищенные данные

- string skillName
- string skillDescription

6.11.1 Подробное описание

См. определение в файле BasePassiveSkill.cs строка 5

6.11.2 Методы

6.11.2.1 ActivatePassiveSkill()

virtual void BasePassiveSkill.ActivatePassiveSkill (
GameObject caster) [inline], [virtual]

См. определение в файле BasePassiveSkill.cs строка 24

```
00024                                           {
00025         Debug.Log(caster.GetComponent<BaseEntity>().GetEntityName() + " activated passive skill");
00026     }
```

6.11.2.2 GetDescription()

string BasePassiveSkill.GetDescription ( ) [inline]

См. определение в файле BasePassiveSkill.cs строка 19

```
00020     {
00021         return skillDescription;
00022     }
```

6.11.2.3 GetName()

string BasePassiveSkill.GetName ( ) [inline]

См. определение в файле BasePassiveSkill.cs строка 14

```
00015     {
00016         return skillName;
00017     }
```

6.11.3 Данные класса

6.11.3.1 skillDescription

string BasePassiveSkill.skillDescription [protected]

См. определение в файле BasePassiveSkill.cs строка 12

6.11.3.2 skillName

string BasePassiveSkill.skillName [protected]

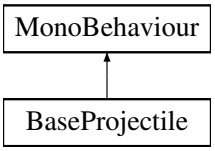
См. определение в файле BasePassiveSkill.cs строка 9

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/BaseObjects/BasePassiveSkill.cs

6.12 Класс BaseProjectile

Граф наследования:BaseProjectile:



Открытые члены

- void SetSenderCollider (GameObject senderObject)
- string GetProjectileName ()
- string GetProjectileDescription ()
- void SetAimDirection (Vector3 newAimDirection)
- void SetAimAngel (float newAimAngel)
- void SetDestroyTags (List< string > newDestroyTags)
- void SetDamageTags (List< string > newDamageTags)
- void AddDamage (int additionalDamage)

6.12.1 Подробное описание

См. определение в файле BaseProjectile.cs строка 7

6.12.2 Методы

6.12.2.1 AddDamage()

void BaseProjectile.AddDamage (
int additionalDamage) [inline]

См. определение в файле BaseProjectile.cs строка 66

```
00067     {
00068         this.projectileDamage += additionalDamage;
00069     }
```

6.12.2.2 GetProjectileDescription()

string BaseProjectile.GetProjectileDescription ( ) [inline]

См. определение в файле BaseProjectile.cs строка 40

```
00041     {
00042         return projectileDescription;
00043     }
```

6.12.2.3 GetProjectileName()

string BaseProjectile.GetProjectileName ( ) [inline]

См. определение в файле BaseProjectile.cs строка 35

```
00036     {
00037         return projectileName;
00038     }
```

6.12.2.4 SetAimAngel()

void BaseProjectile.SetAimAngel (
float newAimAngel) [inline]

См. определение в файле BaseProjectile.cs строка 50

```
00051     {
00052         aimAngle = newAimAngel;
00053     }
```

6.12.2.5 SetAimDirection()

void BaseProjectile.SetAimDirection (
Vector3 newAimDirection) [inline]

См. определение в файле BaseProjectile.cs строка 45

```
00046     {
00047         aimDirection = newAimDirection;
00048     }
```

6.12.2.6 SetDamageTags()

void BaseProjectile.SetDamageTags (
List< string > newDamageTags) [inline]

См. определение в файле BaseProjectile.cs строка 61

```
00062     {
00063         _damageTags = newDamageTags;
00064     }
```

6.12.2.7 SetDestroyTags()

void BaseProjectile.SetDestroyTags (
List< string > newDestroyTags) [inline]

См. определение в файле BaseProjectile.cs строка 55

```
00056     {
00057         _destroyTags = newDestroyTags;
00058     }
```

6.12.2.8 SetSenderCollider()

```
void BaseProjectile.SetSenderCollider (
                GameObject senderObject )   [inline]
```

См. определение в файле BaseProjectile.cs строка 30

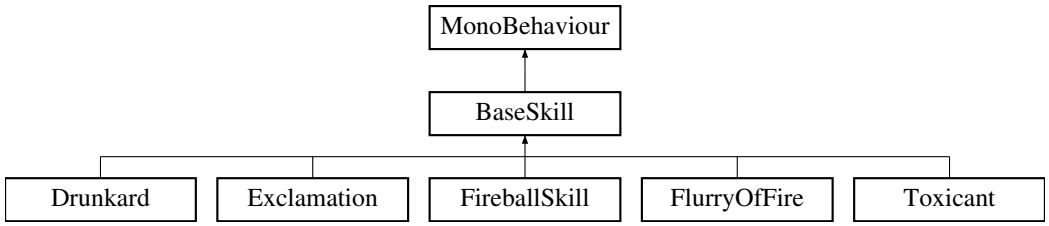
```
00031     {
00032         senderGameObject = senderObject;
00033     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/BaseObjects/BaseProjectile.cs

6.13 Класс BaseSkill

Граф наследования:BaseSkill:



Открытые члены

- string GetName ()
- string GetDescription ()
- int GetCost ()
- float GetCastTime ()
- float GetCooldownTime ()
- Sprite GetSprite ()
- virtual IEnumerator UseSkill (GameObject caster, string key)

Открытые атрибуты

- Action< float > onRelease

Статические открытые данные

- static Action< float > onCast

Защищенные данные

- string skillName
- string skillDescription
- int energyCost
- float castTime
- float cooldown
- bool cancelMovementOnCast
- Sprite skillSprite

6.13.1 Подробное описание

См. определение в файле BaseSkill.cs строка 7

6.13.2 Методы

6.13.2.1 GetCastTime()

float BaseSkill.GetCastTime ( ) [inline]

См. определение в файле BaseSkill.cs строка 44

```
00045     {
00046         return castTime;
00047     }
```

6.13.2.2 GetCooldownTime()

float BaseSkill.GetCooldownTime ( ) [inline]

См. определение в файле BaseSkill.cs строка 49

```
00050     {
00051         return cooldown;
00052     }
```

6.13.2.3 GetCost()

int BaseSkill.GetCost ( ) [inline]

См. определение в файле BaseSkill.cs строка 39

```
00040     {
00041         return energyCost;
00042     }
```

6.13.2.4 GetDescription()

string BaseSkill.GetDescription ( ) [inline]

См. определение в файле BaseSkill.cs строка 35

```
00036     {
00037         return skillDescription;
00038     }
```

6.13.2.5 GetName()

string BaseSkill.GetName ( ) [inline]

См. определение в файле BaseSkill.cs строка 30

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

6.13.2.6 GetSprite()

Sprite BaseSkill.GetSprite ( ) [inline]

См. определение в файле BaseSkill.cs строка 54

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

6.13.2.7 UseSkill()

virtual IEnumerator BaseSkill.UseSkill (
GameObject caster,
string key) [inline], [virtual]

См. определение в файле BaseSkill.cs строка 62

```
00062                                                 {
00063         yield return new WaitForSeconds(0.0f);
00064     }
```

6.13.3 Данные класса

6.13.3.1 cancelMovementOnCast

bool BaseSkill.cancelMovementOnCast [protected]

См. определение в файле BaseSkill.cs строка 25

6.13.3.2 castTime

float BaseSkill.castTime [protected]

См. определение в файле BaseSkill.cs строка 19

6.13.3.3 cooldown

float BaseSkill.cooldown [protected]

См. определение в файле BaseSkill.cs строка 22

6.13.3.4 energyCost

int BaseSkill.energyCost [protected]

См. определение в файле BaseSkill.cs строка 16

6.13.3.5 onCast

Action<float> BaseSkill.onCast [static]

См. определение в файле BaseSkill.cs строка 59

6.13.3.6 onRelease

Action<float> BaseSkill.onRelease

См. определение в файле BaseSkill.cs строка 60

6.13.3.7 skillDescription

string BaseSkill.skillDescription [protected]

См. определение в файле BaseSkill.cs строка 13

6.13.3.8 skillName

string BaseSkill.skillName [protected]

См. определение в файле BaseSkill.cs строка 10

6.13.3.9 skillSprite

Sprite BaseSkill.skillSprite [protected]

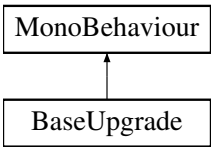
См. определение в файле BaseSkill.cs строка 28

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/BaseObjects/BaseSkill.cs

6.14 Класс BaseUpgrade

Граф наследования:BaseUpgrade:



Открытые члены

- string GetName ()
- string GetDescription ()
- int GetCost ()
- void ApplyUpgrade (BaseEntity player)

6.14.1 Подробное описание

См. определение в файле BaseUpgrade.cs строка 5

6.14.2 Методы

6.14.2.1 ApplyUpgrade()

void BaseUpgrade.ApplyUpgrade (
BaseEntity player) [inline]

См. определение в файле BaseUpgrade.cs строка 24

```
00025     {
00026         Debug.Log("Upgrade " + upgradeName + " applied to " + player.GetEntityName());
00027     }
```

6.14.2.2 GetCost()

int BaseUpgrade.GetCost ( ) [inline]

См. определение в файле BaseUpgrade.cs строка 19

```
00020     {
00021         return upgradeCost;
00022     }
```

6.14.2.3 GetDescription()

string BaseUpgrade.GetDescription ( ) [inline]

См. определение в файле BaseUpgrade.cs строка 15

```
00016     {
00017         return upgradeDescription;
00018     }
```

6.14.2.4 GetName()

string BaseUpgrade.GetName ( ) [inline]

См. определение в файле BaseUpgrade.cs строка 11

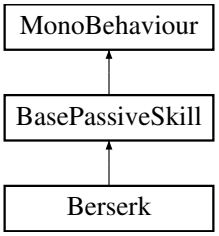
```
00012     {
00013         return upgradeName;
00014     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/BaseObjects/BaseUpgrade.cs

6.15 Класс Berserk

Граф наследования:Berserk:



Дополнительные унаследованные члены

6.15.1 Подробное описание

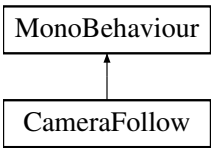
См. определение в файле Berserk.cs строка 5

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/PassiveSkills/Berserk.cs

6.16 Класс CameraFollow

Граф наследования:CameraFollow:



Открытые атрибуты

- Transform player

6.16.1 Подробное описание

См. определение в файле CameraFollow.cs строка 5

6.16.2 Данные класса

6.16.2.1 player

Transform CameraFollow.player

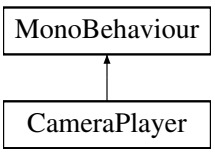
См. определение в файле CameraFollow.cs строка 7

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Camera/CameraFollow.cs

6.17 Класс CameraPlayer

Граф наследования:CameraPlayer:



6.17.1 Подробное описание

См. определение в файле CameraPlayer.cs строка 4

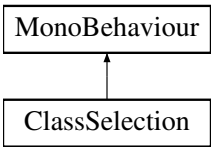
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Player/CameraPlayer.cs

6.18 Класс ClassSelection

Модуль управления панелью выбора класса пользователя

Граф наследования:ClassSelection:



6.18.1 Подробное описание

Модуль управления панелью выбора класса пользователя

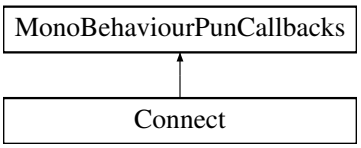
См. определение в файле ClassSelection.cs строка 11

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Player/ClassSelection.cs

6.19 Класс Connect

Граф наследования:Connect:



Открытые члены

- override void OnConnectedToMaster ()

6.19.1 Подробное описание

См. определение в файле Connect.cs строка 3

6.19.2 Методы

6.19.2.1 OnConnectedToMaster()

override void Connect.OnConnectedToMaster ( ) [inline]

См. определение в файле Connect.cs строка 16

```
00017     {
00018         // проверка на работу fastapi сервера и авто-авторизация
00019         _controllers.CheckVujiServer();
00020     }
```

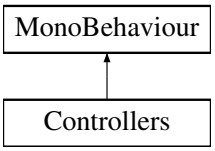
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Loading/Connect.cs

6.20 Класс Controllers

Класс для взаимодействий с сервером Vuji

Граф наследования:Controllers:



Открытые члены

- void CheckVujiServer ()

  Проверяет, доступен ли сервер Vuji
- void Login (string login, string password)

  Получает токен пользователя с сервера Vuji по указанным данным
- void Register (string login, string password)

  Создает аккаунт пользователя по указанным данным
- void UserOnline ()

  Сообщает серверу о том, что пользователь в сети
- void UserOffline ()

  Сообщает серверу о том, что пользователь вышел из аккаунта
- void GetUserID (string token)

  Запрашивает ID пользователя по токену авторизации (из базы данных)
- void FindFriendsByName (string friendsName)

  Поиск пользователя по имени
- void SetLocalUserName (Text field)

  Установить значение текстового поля на имя локального пользователя

6.20.1 Подробное описание

Класс для взаимодействий с сервером Vuji

См. определение в файле Controllers.cs строка 13

6.20.2 Методы

6.20.2.1 CheckVujiServer()

void Controllers.CheckVujiServer ( ) [inline]

Проверяет, доступен ли сервер Vuji

См. определение в файле Controllers.cs строка 31

```
00032     {
00033         StartCoroutine(CheckVujiServerNet());
00034     }
```

6.20.2.2 FindFriendsByName()

void Controllers.FindFriendsByName (
string friendsName) [inline]

Поиск пользователя по имени

Аргументы

friendsName	Имя пользователя

См. определение в файле Controllers.cs строка 82

```
00083     {
00084         string token = _dataBase.GetToken();
00085         StartCoroutine(FindFriendsByNameNet(token, friendsName));
00086     }
```

6.20.2.3 GetUserID()

void Controllers.GetUserID (
string token) [inline]

Запрашивает ID пользователя по токену авторизации (из базы данных)

Аргументы

token	Токен авторизации пользователя

См. определение в файле Controllers.cs строка 74

```
00075     {
00076         StartCoroutine(GetUserIDNet(token));
00077     }
```

6.20.2.4 Login()

```
void Controllers.Login (
                string login,
                string password )    [inline]
```

Получает токен пользователя с сервера Vuji по указанным данным

Аргументы

login	Логин пользователя
password	Пароль пользователя

См. определение в файле Controllers.cs строка 40

```
00041     {
00042         StartCoroutine(LoginNet(login, password));
00043     }
```

6.20.2.5 Register()

```
void Controllers.Register (
                string login,
                string password )    [inline]
```

Создает аккаунт пользователя по указанным данным

Аргументы

login	Логин пользователя
password	Пароль пользователя

См. определение в файле Controllers.cs строка 49

```
00050     {
00051         StartCoroutine(RegisterNet(login, password));
00052     }
```

6.20.2.6 SetLocalUserName()

void Controllers.SetLocalUserName (
Text field) [inline]

Установить значение текстового поля на имя локального пользователя

Аргументы

field	Текстовое поле, значение которого необходимо изменить

См. определение в файле Controllers.cs строка 91

```
00092     {
00093         if (_dataBase == null) _dataBase = gameObject.GetComponent<DataBase>(); ;
00094         string token = _dataBase.GetToken();
00095         StartCoroutine(GetUserInfo(token, field));
00096
00097
00098     }
```

6.20.2.7 UserOffline()

void Controllers.UserOffline ( ) [inline]

Сообщает серверу о том, что пользователь вышел из аккаунта

См. определение в файле Controllers.cs строка 64

```
00065     {
00066         string token = _dataBase.GetToken();
00067         _dataBase.SetToken("");
00068         StartCoroutine(UserOfflineNet(token));
00069     }
```

6.20.2.8 UserOnline()

void Controllers.UserOnline ( ) [inline]

Сообщает серверу о том, что пользователь в сети

См. определение в файле Controllers.cs строка 56

```
00057     {
00058         string token = _dataBase.GetToken();
00059         StartCoroutine(UserOnlineNet(token));
00060     }
```

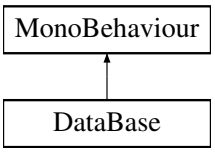
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Controllers.cs

6.21 Класс DataBase

Модуль взаимодействия с базой данных игры

Граф наследования:DataBase:



Открытые члены

- string GetToken ()
  Метод возвращает токен из БД
- void SetToken (string token)
  Метод записывает новый токен в БД
- bool ExistKeybind (string name)
  Проверяет, существует ли в базе данных настройка управления по названию действия
- void AddKeybind (string name, KeyCode key, string category)
  Добавляет настройку управления в базу данных с указанными параметрами
- void SetKeybind (string name, KeyCode key)
  Установить ключ действия в базе данных по названию
- List< Keybind > GetKeybinds ()
  Получить список настоек управления из базы данных
- void AddSetting (string name, string value)
  Добавить настроку в базу данных с указанными параметрами
- bool ExistSetting (string name)
  Проверить, существует ли настройка с указаным именем в базе данных
- void SetSetting (string name, string value)
  Установить значение для настройки с указанным именем
- List< Setting > GetSettings ()
  Получить список настроек из базы данных

6.21.1 Подробное описание

Модуль взаимодействия с базой данных игры

См. определение в файле DataBase.cs строка 9

6.21.2 Методы

6.21.2.1 AddKeybind()

```
void DataBase.AddKeybind (
            string name,
            KeyCode key,
            string category )   [inline]
```

Добавляет настройку управления в базу данных с указанными параметрами

Аргументы

name	Название действия
key	Ключ действия
category	Категория действия ("Movement", "Ability", "UI")

См. определение в файле DataBase.cs строка 187

```
00188     {
00189         OpenConnection();
00190         _command.CommandText = "INSERT INTO keybinds (name, keybind, category) VALUES ('" + name + "', '" +
      key.ToString() + "', '" + category + "');";
00191         _command.ExecuteNonQuery();
00192         CloseConnection();
00193     }
```

6.21.2.2 AddSetting()

```
void DataBase.AddSetting (
                string name,
                string value )   [inline]
```

Добавить настроку в базу данных с указанными параметрами

Аргументы

name	Название настройки
value	Значение настройки

См. определение в файле DataBase.cs строка 229

```
00230     {
00231         OpenConnection();
00232         _command.CommandText = "INSERT INTO settings (name, value) VALUES ('" + name + "', '" + value + "');";
00233         _command.ExecuteNonQuery();
00234         CloseConnection();
00235     }
```

6.21.2.3 ExistKeybind()

```
bool DataBase.ExistKeybind (
                string name )   [inline]
```

Проверяет, существует ли в базе данных настройка управления по названию действия

Аргументы

name	Название действия

Возвращает

Существует ли настройка в базе данных

См. определение в файле DataBase.cs строка 167

```
00168     {
00169         OpenConnection();
00170         _command.CommandText = "SELECT * from keybinds WHERE name ='" + name + "';";
00171         _reader = _command.ExecuteReader();
00172         if (_reader.Read())
00173         {
00174             CloseConnection();
00175             return true;
00176         }
00177         CloseConnection();
00178         return false;
00179
00180     }
```

6.21.2.4 ExistSetting()

bool DataBase.ExistSetting (
string name) [inline]

Проверить, существует ли настройка с указаным именем в базе данных

Аргументы

name	Имя настройки

Возвращает

Существует ли настройка

См. определение в файле DataBase.cs строка 241

```
00242     {
00243         OpenConnection();
00244         _command.CommandText = "SELECT * from settings WHERE name ='" + name + "';";
00245         _reader = _command.ExecuteReader();
00246         if (_reader.Read())
00247         {
00248             CloseConnection();
00249             return true;
00250         }
00251         CloseConnection();
00252         return false;
00253     }
```

6.21.2.5 GetKeybinds()

List< Keybind > DataBase.GetKeybinds ( ) [inline]

Получить список настоек управления из базы данных

Возвращает

Список настроек управления

См. определение в файле DataBase.cs строка 211

```
00212     {
00213         List<Keybind> keybinds = new List<Keybind>();
00214         OpenConnection();
00215         _command.CommandText = "SELECT * from keybinds";
00216         _reader = _command.ExecuteReader();
00217         while (_reader.Read())
00218         {
00219             keybinds.Add(new Keybind(_reader["name"].ToString(), _reader["keybind"].ToString(),
      _reader["category"].ToString()));
00220         }
00221         CloseConnection();
00222         return keybinds;
00223     }
```

6.21.2.6 GetSettings()

List< Setting > DataBase.GetSettings ( ) [inline]

Получить список настроек из базы данных

Возвращает

Список настроек

См. определение в файле DataBase.cs строка 271

```
00272     {
00273         List<Setting> keybinds = new List<Setting>();
00274         OpenConnection();
00275         _command.CommandText = "SELECT * from settings";
00276         _reader = _command.ExecuteReader();
00277         while (_reader.Read())
00278         {
00279             keybinds.Add(new Setting(_reader["name"].ToString(), _reader["value"].ToString()));
00280         }
00281         CloseConnection();
00282         return keybinds;
00283     }
```

6.21.2.7 GetToken()

string DataBase.GetToken ( ) [inline]

Метод возвращает токен из БД

Возвращает

токен

См. определение в файле DataBase.cs строка 117

```
00118     {
00119         string token = "None";
00120         if (TokenInDB())
00121         {
00122             OpenConnection();
00123             _command.CommandText = "SELECT * FROM token WHERE id=1";
00124             _reader = _command.ExecuteReader();
00125             while (_reader.Read())
00126             {
00127                 token = _reader["token"].ToString();
00128             }
00129 
00130             CloseConnection();
00131         }
00132 
00133         if (token == "")
00134         {
00135             token = "None";
00136         }
00137 
00138         return token;
00139     }
```

6.21.2.8 SetKeybind()

void DataBase.SetKeybind (
string name,
KeyCode key) [inline]

Установить ключ действия в базе данных по названию

Аргументы

name	Название действия
key	Новый ключ действия

См. определение в файле DataBase.cs строка 199

```
00200     {
00201         if (!ExistKeybind(name)) { return; }
00202         OpenConnection();
00203         _command.CommandText = "UPDATE keybinds SET keybind  = '" + key.ToString() + "' WHERE name ='" +
      name + "';";
00204         _command.ExecuteNonQuery();
00205         CloseConnection();
00206     }
```

6.21.2.9 SetSetting()

void DataBase.SetSetting (
string name,
string value) [inline]

Установить значение для настройки с указанным именем

Аргументы

name	Название настройки
value	Новое значение

См. определение в файле DataBase.cs строка 259

```
00260     {
00261         if (!ExistSetting(name)) { return; }
00262         OpenConnection();
00263         _command.CommandText = "UPDATE settings SET value  = '" + value + "' WHERE name ='" + name + "';";
00264         _command.ExecuteNonQuery();
00265         CloseConnection();
00266     }
```

6.21.2.10 SetToken()

void DataBase.SetToken (
string token) [inline]

Метод записывает новый токен в БД

Аргументы

token	новый токен

См. определение в файле DataBase.cs строка 145

```
00146     {
00147         if (TokenInDB())
00148         {
00149             OpenConnection();
00150             _command.CommandText = "UPDATE token SET token = '" + token + "' WHERE id =1;";
00151             _command.ExecuteNonQuery();
00152             CloseConnection();
00153         }
00154         else
00155         {
00156             OpenConnection();
00157             _command.CommandText = "INSERT INTO token (token) VALUES ('" + token + "');";
00158             _command.ExecuteNonQuery();
00159             CloseConnection();
00160         }
00161     }
```

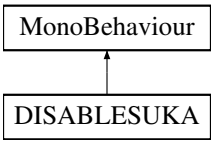
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/DataBase.cs

6.22 Класс DISABLESUKA

Вспомогательный скрипт, выключает все объекты (список задается в редакторе юнити)

Граф наследования:DISABLESUKA:



6.22.1 Подробное описание

Вспомогательный скрипт, выключает все объекты (список задается в редакторе юнити)

См. определение в файле DISABLESUKA.cs строка 7

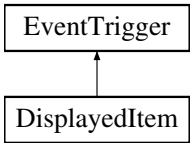
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/DISABLESUKA.cs

6.23 Класс DisplayedItem

Модуль для управления отображаемым предметом инвентаря

Граф наследования:DisplayedItem:



Открытые члены

- override void OnPointerDown (PointerEventData eventData)
  При зажатии курсором панели предмета
- override void OnPointerUp (PointerEventData eventData)
  При отжатии курсора

Открытые атрибуты

- RectTransform inventoryPanel
- RectTransform displayedItem
- int itemId
- Vector2 itemPosition

Статические открытые данные

- static Action< int > onItemDrop
- static Action< DisplayedItem > onItemSwap

6.23.1 Подробное описание

Модуль для управления отображаемым предметом инвентаря

См. определение в файле DisplayedItem.cs строка 10

6.23.2 Методы

6.23.2.1 OnPointerDown()

override void DisplayedItem.OnPointerDown (
PointerEventData eventData) [inline]

При зажатии курсором панели предмета

Аргументы

eventData	

См. определение в файле DisplayedItem.cs строка 42

```
00043     {
00044         itemPosition = displayedItem.position;
00045         isDragging = true;
00046     }
```

6.23.2.2 OnPointerUp()

override void DisplayedItem.OnPointerUp (
PointerEventData eventData) [inline]

При отжатии курсора

Аргументы

eventData	

См. определение в файле DisplayedItem.cs строка 51

```
00052     {
00053         isDragging = false;
00054         if (!RectTransformUtility.RectangleContainsScreenPoint(inventoryPanel, Input.mousePosition))
00055         {
00056             onItemDrop?.Invoke(itemId);
00057         }
00058         else
00059         {
00060             onItemSwap?.Invoke(this);
00061         }
00062     }
```

6.23.3 Данные класса

6.23.3.1 displayedItem

RectTransform DisplayedItem.displayedItem

См. определение в файле DisplayedItem.cs строка 13

6.23.3.2 inventoryPanel

RectTransform DisplayedItem.inventoryPanel

См. определение в файле DisplayedItem.cs строка 12

6.23.3.3 itemId

int DisplayedItem.itemId

См. определение в файле DisplayedItem.cs строка 14

6.23.3.4 itemPosition

Vector2 DisplayedItem.itemPosition

См. определение в файле DisplayedItem.cs строка 15

6.23.3.5 onItemDrop

Action<int> DisplayedItem.onItemDrop [static]

См. определение в файле DisplayedItem.cs строка 17

6.23.3.6 onItemSwap

Action<DisplayedItem> DisplayedItem.onItemSwap [static]

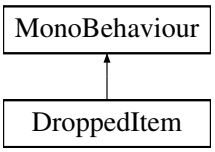
См. определение в файле DisplayedItem.cs строка 18

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Inventory/DisplayedItem.cs

6.24 Класс DroppedItem

Граф наследования:DroppedItem:



Открытые члены

- void SetItem (BaseItem aitemData)

Открытые атрибуты

- BaseItem itemData

6.24.1 Подробное описание

См. определение в файле DroppedItem.cs строка 3

6.24.2 Методы

6.24.2.1 SetItem()

```
void DroppedItem.SetItem (
                BaseItem aitemData )   [inline]
```

См. определение в файле DroppedItem.cs строка 14

```
00015     {
00016         _spriteRenderer = GetComponent<SpriteRenderer>();
00017         this.itemData = aitemData;
00018         _spriteRenderer.sprite = aitemData.GetImage();
00019     }
```

6.24.3 Данные класса

6.24.3.1 itemData

```
BaseItem DroppedItem.itemData
```

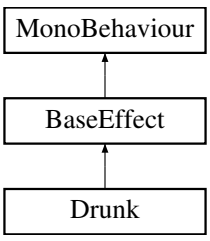
См. определение в файле DroppedItem.cs строка 5

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Items/DroppedItem.cs

6.25 Класс Drunk

Граф наследования:Drunk:



Открытые члены

- override void ApplyEffect (GameObject entity)
- IEnumerator DrunkEffect (GameObject entity)

Открытые атрибуты

- int additionalDefense

6.25.1 Подробное описание

См. определение в файле Drunk.cs строка 5

6.25.2 Методы

6.25.2.1 ApplyEffect()

override void Drunk.ApplyEffect (
GameObject entity) [inline], [virtual]

Переопределяет метод предка BaseEffect.

См. определение в файле Drunk.cs строка 10

```
00011     {
00012         base.ApplyEffect(entity);
00013         entity.gameObject.GetComponent<MonoBehaviour>().StartCoroutine(DrunkEffect(entity));
00014     }
```

6.25.2.2 DrunkEffect()

IEnumerator Drunk.DrunkEffect (
GameObject entity) [inline]

См. определение в файле Drunk.cs строка 16

```
00016                                       {
00017         entity.GetComponent<BaseEntity>().IncreaseDefense(additionalDefense);
00018         yield return new WaitForSeconds(duration);
00019         entity.GetComponent<BaseEntity>().DecreaseDefense(additionalDefense);
00020     }
```

6.25.3 Данные класса

6.25.3.1 additionalDefense

int Drunk.additionalDefense

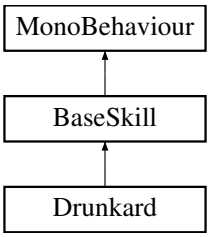
См. определение в файле Drunk.cs строка 8

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Effects/Drunk.cs

6.26 Класс Drunkard

Граф наследования:Drunkard:



Открытые члены

- override IEnumerator UseSkill (GameObject caster, string key)

Дополнительные унаследованные члены

6.26.1 Подробное описание

См. определение в файле Drunkard.cs строка 5

6.26.2 Методы

6.26.2.1 UseSkill()

```
override IEnumerator Drunkard.UseSkill (
                GameObject caster,
                string key )   [inline], [virtual]
```

Переопределяет метод предка BaseSkill.

См. определение в файле Drunkard.cs строка 15

```
00016     {
00017         base.UseSkill(caster, key);
00018         if (caster.GetComponent<BaseEntity>().spendEnergy(energyCost) == false)
00019         {
00020             Debug.Log("Not enough energy to cast " + this.gameObject.name);
00021             yield break;
00022         }
00023 
00024         onCast?.Invoke(castTime);
00025         caster.GetComponent<BaseEntity>().setIsCooldown(key, true);
00026         if(cancelMovementOnCast)
00027             caster.GetComponent<MovementPlayer>().cancelMovement(castTime);
00028 
00029         AnimationPlayer anim = caster.GetComponent<AnimationPlayer>();
00030         anim.ChangePlayerAnimation_q(anim._drink);
00031         yield return new WaitForSeconds(castTime);
00032 
00033         // Сам скилл
00034         drunkEffect.GetComponent<Drunk>().additionalDefense = additionalDefense;
00035         drunkEffect.GetComponent<Drunk>().duration = drunkTime;
00036         onRelease?.Invoke(cooldown);
00037 
00038         caster.GetComponent<BaseEntity>().AddEffect(drunkEffect);
00039         // Сам скилл
00040 
00041         yield return new WaitForSeconds(cooldown);
00042         caster.GetComponent<BaseEntity>().setIsCooldown(key, false);
00043     }
```

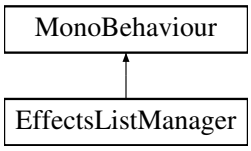
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Skills/Drunkard.cs

6.27 Класс EffectsListManager

Модуль по управлению списком эффектов

Граф наследования:EffectsListManager:



6.27.1 Подробное описание

Модуль по управлению списком эффектов

См. определение в файле EffectsListManager.cs строка 7

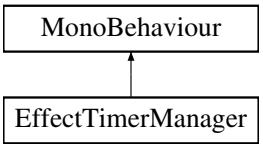
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/Managers/EffectsListManager.cs

6.28 Класс EffectTimerManager

Класс для управления таймером эффекта

Граф наследования:EffectTimerManager:



Открытые члены

- void SetEffect (BaseEffect effect)
  Установка эффекта на данных

6.28.1 Подробное описание

Класс для управления таймером эффекта

См. определение в файле EffectTimerManager.cs строка 9

6.28.2 Методы

6.28.2.1 SetEffect()

void EffectTimerManager.SetEffect (
BaseEffect effect) [inline]

Установка эффекта на данных

Аргументы

effect	Целевой эффект

См. определение в файле EffectTimerManager.cs строка 31

```
00032     {
00033         targetedSprite.sprite = effect.effectSprite;
00034         timerInterval = effect.duration;
00035         targetedEffect = effect;
00036         tooltipText.text = "\"" + effect.effectName + "\"\n" + effect.description + "\n" + "Duration: " + effect.duration
      + "s";
00037     }
```

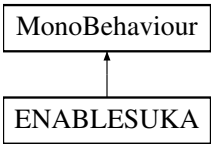
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/Managers/EffectTimerManager.cs

6.29 Класс ENABLESUKA

Вспомогательный скрипт, включает все объекты (список задается в редакторе юнити)

Граф наследования:ENABLESUKA:



6.29.1 Подробное описание

Вспомогательный скрипт, включает все объекты (список задается в редакторе юнити)

См. определение в файле ENABLESUKA.cs строка 7

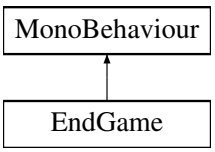
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/ENABLESUKA.cs

6.30 Класс EndGame

Модуль для отслеживания окончания игры

Граф наследования:EndGame:



Открытые члены

- void TeamOneWin ()
- void TeamTwoWin ()

Статические открытые данные

- static Action< string > OnGameEnd

6.30.1 Подробное описание

Модуль для отслеживания окончания игры

См. определение в файле EndGame.cs строка 6

6.30.2 Методы

6.30.2.1 TeamOneWin()

void EndGame.TeamOneWin ( ) [inline]

См. определение в файле EndGame.cs строка 15

```
00016     {
00017         Debug.Log("TEAM ONE WIN");
00018         OnGameEnd?.Invoke("TeamOne");
00019     }
```

6.30.2.2 TeamTwoWin()

void EndGame.TeamTwoWin ( ) [inline]

См. определение в файле EndGame.cs строка 20

```
00021     {
00022         Debug.Log("TEAM TWO WIN");
00023         OnGameEnd?.Invoke("TeamTwo");
00024     }
```

6.30.3 Данные класса

6.30.3.1 OnGameEnd

Action<string> EndGame.OnGameEnd [static]

См. определение в файле EndGame.cs строка 8

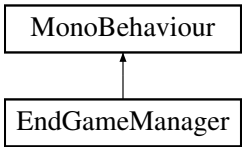
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/EndGame.cs

6.31 Класс EndGameManager

Вспомогательный модуль для отображения панели после окончания игры

Граф наследования:EndGameManager:



Открытые члены

- void LeaveGame ()
  Вспомогательная функция для кнопки выхода из игры

6.31.1 Подробное описание

Вспомогательный модуль для отображения панели после окончания игры

См. определение в файле EndGameManager.cs строка 12

6.31.2 Методы

6.31.2.1 LeaveGame()

void EndGameManager.LeaveGame ( ) [inline]

Вспомогательная функция для кнопки выхода из игры

См. определение в файле EndGameManager.cs строка 60

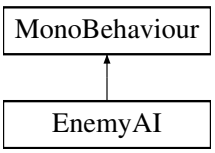
```
00061     {
00062         Destroy(UiCanvas);
00063         Destroy(GameManager);
00064         SceneManager.LoadScene("Lobby");
00065     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/Managers/EndGameManager.cs

6.32 Класс EnemyAI

Граф наследования:EnemyAI:



Открытые члены

- void AgressionStart (GameObject target)

Открытые атрибуты

- bool reachedEndOfPath = false

6.32.1 Подробное описание

См. определение в файле EnemyAI.cs строка 6

6.32.2 Методы

6.32.2.1 AgressionStart()

```
void EnemyAI.AgressionStart (
                GameObject target )   [inline]
```

См. определение в файле EnemyAI.cs строка 24

```
00025     {
00026         this.target = target;
00027         InvokeRepeating("UpdatePath", 0f, 0.5f);
00028     }
```

6.32.3 Данные класса

6.32.3.1 reachedEndOfPath

```
bool EnemyAI.reachedEndOfPath = false
```

См. определение в файле EnemyAI.cs строка 13

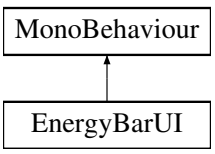
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/AI/EnemyAI.cs

6.33 Класс EnergyBarUI

Модуль управления баром энергии в нижней части экрана

Граф наследования:EnergyBarUI:



Открытые члены

- void SetEntity (BaseEntity player)

6.33.1 Подробное описание

Модуль управления баром энергии в нижней части экрана

См. определение в файле EnergyBarUI.cs строка 8

6.33.2 Методы

6.33.2.1 SetEntity()

```
void EnergyBarUI.SetEntity (
                BaseEntity player )   [inline]
```

См. определение в файле EnergyBarUI.cs строка 53

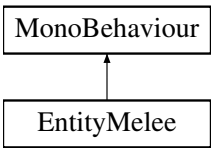
```
00054     {
00055         entity = player;
00056     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/Units/EnergyBarUI.cs

6.34 Класс EntityMelee

Граф наследования:EntityMelee:



Открытые члены

- void Attack (GameObject target)

6.34.1 Подробное описание

См. определение в файле EntityMelee.cs строка 5

6.34.2 Методы

6.34.2.1 Attack()

```
void EntityMelee.Attack (
                GameObject target )   [inline]
```

См. определение в файле EntityMelee.cs строка 21

```
00022     {
00023         if (!_isTimeout)
00024         {
00025             StartCoroutine("AttackTiemout");
00026 
00027             var xLen = target.transform.position.x - transform.position.x;
00028             var yLen = target.transform.position.y - transform.position.y;
00029             var xyLen = (float) (Mathf.Sqrt(Mathf.Pow(xLen, 2) + Mathf.Pow(yLen, 2)));
00030             var x = (xLen * attackDistance) / xyLen + transform.position.x;
00031             var y = (yLen * attackDistance) / xyLen + transform.position.y;
00032 
00033             _attackPoint = new Vector3(x, y, 0);
00034 
00035             Collider2D[] hitEnemies = Physics2D.OverlapCircleAll(_attackPoint, attackRange, enemyLayers);
00036 
00037             foreach (Collider2D enemy in hitEnemies)
00038             {
00039                 GameObject enemyGameObject = enemy.transform.parent.gameObject;
00040                 if (enemyGameObject != gameObject)
00041                     enemyGameObject.GetComponent<BaseEntity>().TakeDamage(_damage);
00042             }
00043         }
00044     }
```

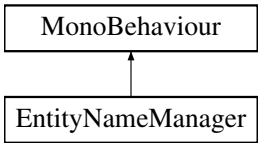
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/AI/EntityMelee.cs

6.35 Класс EntityNameManager

Модуль для упраления и корректировки позиции имени над сущностью

Граф наследования:EntityNameManager:



Открытые члены

- void SetOffset (Vector3 newOffset)

  Изменение сохраненного отклонения от позиции сущности
- Vector3 GetOffset ()

Открытые атрибуты

- Text entityName

6.35.1 Подробное описание

Модуль для упраления и корректировки позиции имени над сущностью

См. определение в файле EntityNameManager.cs строка 9

6.35.2 Методы

6.35.2.1 GetOffset()

Vector3 EntityNameManager.GetOffset ( ) [inline]

Возвращает

Сохраненное отклонение от позиции сущности

См. определение в файле EntityNameManager.cs строка 40

```
00041     {
00042         return offset;
00043     }
```

6.35.2.2 SetOffset()

void EntityNameManager.SetOffset (
Vector3 newOffset) [inline]

Изменение сохраненного отклонения от позиции сущности

Аргументы

newOffset	Новое отклонение

См. определение в файле EntityNameManager.cs строка 32

```
00033     {
00034         offset = newOffset;
00035     }
```

6.35.3 Данные класса

6.35.3.1 entityName

Text EntityNameManager.entityName

См. определение в файле EntityNameManager.cs строка 11

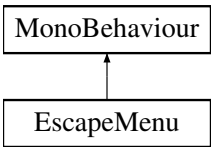
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/Managers/EntityNameManager.cs

6.36 Класс EscapeMenu

Модуль для управления меню паузы

Граф наследования:EscapeMenu:



Открытые члены

- void LeaveLobby ()

  Вспомогательная функция для кнопки выхода из комнаты
- void OpenSettings ()

  Вспомогательная функция для открытия настроек игры
- void ResumeGame ()

  Вспомогательная функция для выхода из меню паузы
- void LeaveGame ()

  Вспомогательная функция для выхода из игры

6.36.1 Подробное описание

Модуль для управления меню паузы

См. определение в файле EscapeMenu.cs строка 9

6.36.2 Методы

6.36.2.1 LeaveGame()

void EscapeMenu.LeaveGame ( ) [inline]

Вспомогательная функция для выхода из игры

См. определение в файле EscapeMenu.cs строка 76

```
00077     {
00078         Application.Quit();
00079     }
```

6.36.2.2 LeaveLobby()

void EscapeMenu.LeaveLobby ( ) [inline]

Вспомогательная функция для кнопки выхода из комнаты

См. определение в файле EscapeMenu.cs строка 51

```
00052     {
00053         PhotonNetwork.LeaveRoom();
00054         SceneManager.LoadScene("Lobby");
00055     }
```

6.36.2.3 OpenSettings()

void EscapeMenu.OpenSettings ( ) [inline]

Вспомогательная функция для открытия настроек игры

См. определение в файле EscapeMenu.cs строка 59

```
00060     {
00061         settingsPanel.gameObject.SetActive(true);
00062         menuPanel.gameObject.SetActive(false);
00063         KeyHandler.instance.SetUIOpened(true);
00064     }
```

6.36.2.4 ResumeGame()

void EscapeMenu.ResumeGame ( ) [inline]

Вспомогательная функция для выхода из меню паузы

См. определение в файле EscapeMenu.cs строка 68

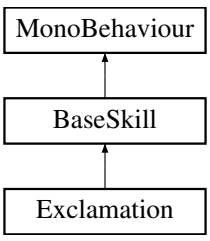
```
00069     {
00070         menuPanel.gameObject.SetActive(false);
00071         KeyHandler.instance.Pause(false);
00072     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/Managers/EscapeMenu.cs

6.37 Класс Exclamation

Граф наследования:Exclamation:



Открытые члены

- override IEnumerator UseSkill (GameObject caster, string key)

Дополнительные унаследованные члены

6.37.1 Подробное описание

См. определение в файле Exclamation.cs строка 5

6.37.2 Методы

6.37.2.1 UseSkill()

```
override IEnumerator Exclamation.UseSkill (
                GameObject caster,
                string key )   [inline], [virtual]
```

Переопределяет метод предка BaseSkill.

См. определение в файле Exclamation.cs строка 10

```
00011     {
00012         base.UseSkill(caster, key);
00013         if (caster.GetComponent<BaseEntity>().spendEnergy(energyCost) == false)
00014         {
00015             Debug.Log("Not enough energy to cast " + this.gameObject.name);
00016             yield break;
00017         }
00018 
00019         onCast?.Invoke(castTime);
00020         caster.GetComponent<BaseEntity>().setIsCooldown(key, true);
00021         if(cancelMovementOnCast)
00022             caster.GetComponent<MovementPlayer>().cancelMovement(castTime);
00023         yield return new WaitForSeconds(castTime);
00024 
00025         // Сам скилл
00026         caster.GetComponent<BaseEntity>().AddEffect(speedEffect);
00027         caster.GetComponent<BaseEntity>().AddEffect(strongEffect);
00028         // Сам скилл
00029         onRelease?.Invoke(cooldown);
00030         yield return new WaitForSeconds(cooldown);
00031         caster.GetComponent<BaseEntity>().setIsCooldown(key, false);
00032     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Skills/Exclamation.cs

6.38 Класс StructsRequest.FindFriendsByNameStructRequest

Открытые атрибуты

- string friendsName

6.38.1 Подробное описание

См. определение в файле Structs.cs строка 24

6.38.2 Данные класса

6.38.2.1 friendsName

string StructsRequest.FindFriendsByNameStructRequest.friendsName

См. определение в файле Structs.cs строка 26

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Structs.cs

6.39 Класс StructsResponse.FindFriendsByNameStructResponse

Открытые атрибуты

- UserInfoObject[] friends

6.39.1 Подробное описание

См. определение в файле Structs.cs строка 53

6.39.2 Данные класса

6.39.2.1 friends

UserInfoObject [] StructsResponse.FindFriendsByNameStructResponse.friends

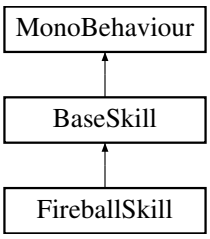
См. определение в файле Structs.cs строка 55

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Structs.cs

6.40 Класс FireballSkill

Граф наследования:FireballSkill:



Открытые члены

- override IEnumerator UseSkill (GameObject caster, string key)

Дополнительные унаследованные члены

6.40.1 Подробное описание

См. определение в файле FireballSkill.cs строка 6

6.40.2 Методы

6.40.2.1 UseSkill()

override IEnumerator FireballSkill.UseSkill (
GameObject caster,
string key) [inline], [virtual]

Переопределяет метод предка BaseSkill.

См. определение в файле FireballSkill.cs строка 10

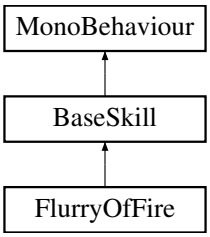
```
00011     {
00012         base.UseSkill(caster, key);
00013 
00014         if (caster.GetComponent<BaseEntity>().spendEnergy(energyCost) == false)
00015         {
00016             Debug.Log("Not enough energy to cast " + this.gameObject.name);
00017             yield break;
00018         }
00019 
00020         onCast?.Invoke(castTime);
00021         caster.GetComponent<BaseEntity>().setIsCooldown(key, true);
00022         if(cancelMovementOnCast)
00023             caster.GetComponent<MovementPlayer>().cancelMovement(castTime);
00024         yield return new WaitForSeconds(castTime);
00025 
00026 
00027         caster.GetComponent<PlayerProjectile>().Attack("Fireball");
00028         onRelease?.Invoke(cooldown);
00029         yield return new WaitForSeconds(cooldown);
00030         caster.GetComponent<BaseEntity>().setIsCooldown(key, false);
00031     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Skills/FireballSkill.cs

6.41 Класс FlurryOfFire

Граф наследования:FlurryOfFire:



Открытые члены

- override IEnumerator UseSkill (GameObject caster, string key)

Дополнительные унаследованные члены

6.41.1 Подробное описание

См. определение в файле FlurryOfFire.cs строка 6

6.41.2 Методы

6.41.2.1 UseSkill()

override IEnumerator FlurryOfFire.UseSkill (
GameObject caster,
string key) [inline], [virtual]

Переопределяет метод предка BaseSkill.

См. определение в файле FlurryOfFire.cs строка 13

```
00014     {
00015         base.UseSkill(caster, key);
00016         if (caster.GetComponent<BaseEntity>().spendEnergy(energyCost) == false)
00017         {
00018             Debug.Log("Not enough energy to cast " + this.gameObject.name);
00019             yield break;
00020         }
00021 
00022         onCast?.Invoke(castTime);
00023         caster.GetComponent<BaseEntity>().setIsCooldown(key, true);
00024         if(cancelMovementOnCast)
00025             caster.GetComponent<MovementPlayer>().cancelMovement(castTime);
00026         yield return new WaitForSeconds(castTime);
00027 
00028         // Сам скилл
00029         for(int i = 0; i < shootTime / timeBetweenShoot; i++)
00030         {
00031             caster.GetComponent<PlayerProjectile>().Attack("Bullet");
00032             yield return new WaitForSeconds(timeBetweenShoot);
00033         }
00034         // Сам скилл
00035         onRelease?.Invoke(cooldown);
00036         yield return new WaitForSeconds(cooldown);
00037         caster.GetComponent<BaseEntity>().setIsCooldown(key, false);
00038     }
```

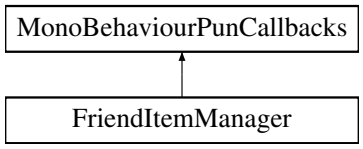
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Skills/FlurryOfFire.cs

6.42 Класс FriendItemManager

Модуль управления объектом пользователя, доступного для приглашения в группу (в лобби)

Граф наследования:FriendItemManager:



Открытые члены

- void InviteFriend ()

  Метод приглашение игрока. Если текущей пользователь не в комнате то создать ее иначе пригласить игрока

Открытые атрибуты

- int userID
- LobbyManager lobbyManager
- Text usernameTextField

6.42.1 Подробное описание

Модуль управления объектом пользователя, доступного для приглашения в группу (в лобби)

См. определение в файле FriendItemManager.cs строка 8

6.42.2 Методы

6.42.2.1 InviteFriend()

void FriendItemManager.InviteFriend ( ) [inline]

Метод приглашение игрока. Если текущей пользователь не в комнате то создать ее иначе пригласить игрока

См. определение в файле FriendItemManager.cs строка 24

```
00025     {
00026         if (PhotonNetwork.InRoom)
00027         {
00028             lobbyManager.CreateInviteFriend(userID, PhotonNetwork.CurrentRoom.Name);
00029         }
00030         else
00031         {
00032             lobbyManager.CreateLobbyAndInviteUser(userID);
00033         }
00034     }
```

6.42.3 Данные класса

6.42.3.1 lobbyManager

LobbyManager FriendItemManager.lobbyManager

См. определение в файле FriendItemManager.cs строка 13

6.42.3.2 userID

int FriendItemManager.userID

См. определение в файле FriendItemManager.cs строка 12

6.42.3.3 usernameTextField

Text FriendItemManager.usernameTextField

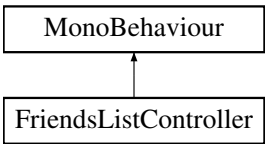
См. определение в файле FriendItemManager.cs строка 14

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/Managers/FriendItemManager.cs

6.43 Класс FriendsListController

Граф наследования:FriendsListController:



Открытые члены

- void OpenFriendsList ()

  Метод открывает/скрывает список друзей
- void FindFriendsByName ()

  Метод для поиск пользователь с похожем именем Затем в корутине будет вызван метод Fill↩ FriendsList из Controllers
- void FillFriendsList (UserInfoObject[] userInfoObjects)

  Заполнение ScrollView префабами найденых пользователей

6.43.1 Подробное описание

См. определение в файле FriendsListController.cs строка 5

6.43.2 Методы

6.43.2.1 FillFriendsList()

void FriendsListController.FillFriendsList (
UserInfoObject[] userInfoObjects) [inline]

Заполнение ScrollView префабами найденых пользователей

Аргументы

userInfoObjects	массив всех найденных пользователей и информации о них.

См. определение в файле FriendsListController.cs строка 59

```
00060     {
00061         // удалить всех старых пользователей
00062         foreach (Transform child in uiFriendListScrollContent)
00063         {
00064             Destroy(child.gameObject);
00065         }
00066 
00067         // добавить новых пользователей
00068         foreach (var userinfo in userInfoObjects)
00069         {
00070             // берем префаб и заполняем его поля
00071             var friendItem = uiFriendItemPrefab.gameObject.GetComponent<FriendItemManager>();
00072             friendItem.userID = userinfo.userID;
00073             friendItem.usernameTextField.text = userinfo.username;
00074             friendItem.lobbyManager = gameObject.GetComponent<LobbyManager>();
00075 
00076             // инициализируем поле установив его родителя как ScrollContent
00077             var instance = Instantiate(friendItem.gameObject);
00078             instance.transform.SetParent(uiFriendListScrollContent.transform, false);
00079         }
00080     }
```

6.43.2.2 FindFriendsByName()

void FriendsListController.FindFriendsByName ( ) [inline]

Метод для поиск пользователь с похожем именем Затем в корутине будет вызван метод FillFriends↩ List из Controllers

См. определение в файле FriendsListController.cs строка 49

```
00050     {
00051         var friendsName = friendsNameInputField.text;
00052         _controllers.FindFriendsByName(friendsName);
00053     }
```

6.43.2.3 OpenFriendsList()

void FriendsListController.OpenFriendsList ( ) [inline]

Метод открывает/скрывает список друзей

См. определение в файле FriendsListController.cs строка 33

```
00034     {
00035         if (uiFriendsList.activeSelf)
00036         {
00037             uiFriendsList.SetActive(false);
00038         }
00039         else
00040         {
00041             uiFriendsList.SetActive(true);
00042         }
00043     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Lobby/FriendsListController.cs

6.44 Класс GameSettings.GameSettingsOriginal

Статические открытые данные

- const int MaxPlayersInGame = 4

6.44.1 Подробное описание

См. определение в файле Structs.cs строка 95

6.44.2 Данные класса

6.44.2.1 MaxPlayersInGame

const int GameSettings.GameSettingsOriginal.MaxPlayersInGame = 4 [static]

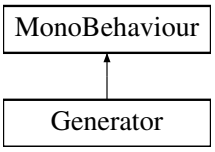
См. определение в файле Structs.cs строка 97

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Structs.cs

6.45 Класс Generator

Граф наследования:Generator:



Открытые атрибуты

- Room[] RoomPrefabs
- Room StartingRoom
- int Width
- int Height

6.45.1 Подробное описание

См. определение в файле Generator.cs строка 7

6.45.2 Данные класса

6.45.2.1 Height

int Generator.Height

См. определение в файле Generator.cs строка 12

6.45.2.2 RoomPrefabs

Room [] Generator.RoomPrefabs

См. определение в файле Generator.cs строка 9

6.45.2.3 StartingRoom

Room Generator.StartingRoom

См. определение в файле Generator.cs строка 10

6.45.2.4 Width

int Generator.Width

См. определение в файле Generator.cs строка 11

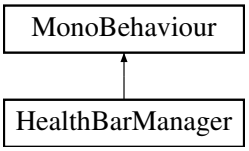
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Map/Generator.cs

6.46 Класс HealthBarManager

Модуль для управления полоской жизней над сущностью

Граф наследования:HealthBarManager:



Открытые члены

- void SetHealth (float health, float maxHp)
  Установить текущие и максимальные хп сущности
- void SetOffset (Vector3 newOffset)
  Установить отклонение от позиции сущности
- Vector3 GetOffset ()

Открытые атрибуты

- Slider slider

6.46.1 Подробное описание

Модуль для управления полоской жизней над сущностью

См. определение в файле HealthBarManager.cs строка 8

6.46.2 Методы

6.46.2.1 GetOffset()

Vector3 HealthBarManager.GetOffset ( ) [inline]

Возвращает

Текущее отклонение от позиции сущности

См. определение в файле HealthBarManager.cs строка 60

```
00061     {
00062         return offset;
00063     }
```

6.46.2.2 SetHealth()

void HealthBarManager.SetHealth (
float health,
float maxHp) [inline]

Установить текущие и максимальные хп сущности

Аргументы

health	Текущее хп
maxHp	Максимальное хп

См. определение в файле HealthBarManager.cs строка 32

```
00033     {
00034         gameObject.SetActive(health < maxHp);
00035         slider.value = health;
00036         slider.maxValue = maxHp;
00037         slider.fillRect.GetComponentInChildren<Image>().color = Color.Lerp(Low, High, slider.normalizedValue);
00038     }
```

6.46.2.3 SetOffset()

void HealthBarManager.SetOffset (
Vector3 newOffset) [inline]

Установить отклонение от позиции сущности

Аргументы

newOffset	Новое отклонение

См. определение в файле HealthBarManager.cs строка 52

```
00053     {
00054         offset = newOffset;
00055     }
```

6.46.3 Данные класса

6.46.3.1 slider

Slider HealthBarManager.slider

См. определение в файле HealthBarManager.cs строка 10

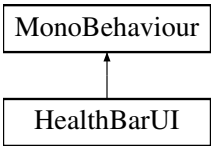
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/Managers/HealthBarManager.cs

6.47 Класс HealthBarUI

Модуль управления баром хп в нижней части экрана

Граф наследования:HealthBarUI:



Открытые члены

- void SetEntity (BaseEntity player)
  Установить целевую сущность

6.47.1 Подробное описание

Модуль управления баром хп в нижней части экрана

См. определение в файле HealthBarUI.cs строка 8

6.47.2 Методы

6.47.2.1 SetEntity()

void HealthBarUI.SetEntity (
BaseEntity player) [inline]

Установить целевую сущность

Аргументы

player	Новая целевая сущность

См. определение в файле HealthBarUI.cs строка 52

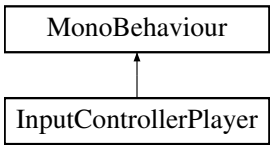
```
00053     {
00054         entity = player;
00055     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/Units/HealthBarUI.cs

6.48 Класс InputControllerPlayer

Граф наследования:InputControllerPlayer:



6.48.1 Подробное описание

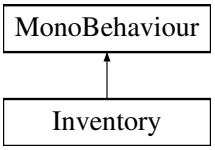
См. определение в файле InputControllerPlayer.cs строка 6

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Player/InputControllerPlayer.cs

6.49 Класс Inventory

Граф наследования:Inventory:



Открытые члены

- void AddItem (BaseItem item, GameObject itemGameObject)
- List< BaseItem > GetAllItems ()
- void ClearInventory ()
- void SyncronisedDrop (int index)
- bool DropItem (int itemId)

Открытые атрибуты

- Action< BaseItem > onItemAdded
- List< BaseItem > inventoryItems = new List<BaseItem>()

6.49.1 Подробное описание

См. определение в файле Inventory.cs строка 7

6.49.2 Методы

6.49.2.1 AddItem()

```
void Inventory.AddItem (
                BaseItem item,
                GameObject itemGameObject )   [inline]
```

См. определение в файле Inventory.cs строка 21

```
00021                                                {
00022         inventoryItems.Add(item);
00023         onItemAdded?.Invoke(item);
00024         Debug.Log("Added item: " + item.GetItemName() + item.GetDescription() +  item.GetAmount());
00025         Destroy(itemGameObject);
00026     }
```

6.49.2.2 ClearInventory()

```
void Inventory.ClearInventory ( )   [inline]
```

См. определение в файле Inventory.cs строка 31

```
00032     {
00033         inventoryItems.Clear();
00034     }
```

6.49.2.3 DropItem()

```
bool Inventory.DropItem (
                int itemId )   [inline]
```

См. определение в файле Inventory.cs строка 48

```
00049     {
00050         _view.RPC("SyncronisedDrop", RpcTarget.All, itemId);
00051
00052         var item = inventoryItems[itemId];
00053         if (item.GetAmount() < 1)
00054         {
00055             return false;
00056         }
00057
00058         return true;
00059     }
```

6.49.2.4 GetAllItems()

List< BaseItem > Inventory.GetAllItems ( ) [inline]

См. определение в файле Inventory.cs строка 28

```
00028                                {
00029         return inventoryItems;
00030     }
```

6.49.2.5 SyncronisedDrop()

void Inventory.SyncronisedDrop (
int index) [inline]

См. определение в файле Inventory.cs строка 38

```
00039     {
00040         var item = inventoryItems[index];
00041         item.SetAmount(item.GetAmount() - 1);
00042 
00043         Vector2 position = new Vector2(gameObject.transform.position.x, gameObject.transform.position.y - 1f);
00044         GameObject droppedItem = Instantiate(_droppedItemPrefab, position, Quaternion.identity);
00045         droppedItem.GetComponent<DroppedItem>().SetItem(item);
00046     }
```

6.49.3 Данные класса

6.49.3.1 inventoryItems

List<BaseItem> Inventory.inventoryItems = new List<BaseItem>()

См. определение в файле Inventory.cs строка 11

6.49.3.2 onItemAdded

Action<BaseItem> Inventory.onItemAdded

См. определение в файле Inventory.cs строка 9

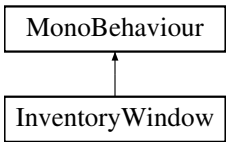
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Inventory/Inventory.cs

6.50 Класс InventoryWindow

Модуль управления отображаемым инвентарем сущности

Граф наследования:InventoryWindow:



Открытые члены

- void Start ()
- void OnSpawn (GameObject playerObject)

6.50.1 Подробное описание

Модуль управления отображаемым инвентарем сущности

См. определение в файле InventoryWindow.cs строка 11

6.50.2 Методы

6.50.2.1 OnSpawn()

void InventoryWindow.OnSpawn (
GameObject playerObject) [inline]

См. определение в файле InventoryWindow.cs строка 38

```
00039     {
00040         player = playerObject;
00041         playerInventory = playerObject.GetComponent<Inventory>();
00042         playerInventory.onItemAdded += OnItemAdded;
00043         Redraw();
00044     }
```

6.50.2.2 Start()

void InventoryWindow.Start ( ) [inline]

См. определение в файле InventoryWindow.cs строка 21

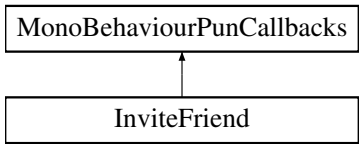
```
00022     {
00023         if (playerInventory != null) playerInventory.onItemAdded += OnItemAdded;
00024         DisplayedItem.onItemDrop += OnItemDropped;
00025         DisplayedItem.onItemSwap += onItemSwapped;
00026         KeyHandler.keyPressed += KeyPressed;
00027         SpawnPlayers.OnSpawn += OnSpawn;
00028     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Inventory/InventoryWindow.cs

6.51 Класс InviteFriend

Граф наследования:InviteFriend:



Открытые члены

- void StartInviteFriend ()

  Метод перед приглашением другого игрока. Тут может быть проверка на статус приглашаемого игрока. Если игрока нет в сети не приглашать его
- override void OnJoinedRoom ()

Открытые атрибуты

- int invitedUserID
- string roomName

6.51.1 Подробное описание

См. определение в файле InviteFriend.cs строка 3

6.51.2 Методы

6.51.2.1 OnJoinedRoom()

override void InviteFriend.OnJoinedRoom ( ) [inline]

См. определение в файле InviteFriend.cs строка 23

```
00024     {
00025         roomName = PhotonNetwork.CurrentRoom.Name;
00026         StartInviteFriend();
00027     }
```

6.51.2.2 StartInviteFriend()

void InviteFriend.StartInviteFriend ( ) [inline]

Метод перед приглашением другого игрока. Тут может быть проверка на статус приглашаемого игрока. Если игрока нет в сети не приглашать его

См. определение в файле InviteFriend.cs строка 13

```
00014     {
00015         if (roomName != null)
00016         {
00017             _socketServerController = GetComponent<SocketServerController>();
00018             _socketServerController.StartSendInviteToSocketServer(invitedUserID, roomName);
00019             enabled = false;
00020         }
00021     }
```

6.51.3 Данные класса

6.51.3.1 invitedUserID

int InviteFriend.invitedUserID

См. определение в файле InviteFriend.cs строка 5

6.51.3.2 roomName

string InviteFriend.roomName

См. определение в файле InviteFriend.cs строка 6

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Lobby/InviteFriend.cs

6.52 Класс Keybind

Вспомогательный класс для передачи информации о сохраненных настройках управления

Открытые члены

- Keybind (string name, string key, string category)
  Конструктор класса

Открытые атрибуты

- string name
- string key
- string category

6.52.1 Подробное описание

Вспомогательный класс для передачи информации о сохраненных настройках управления

См. определение в файле DataBase.cs строка 291

6.52.2 Конструктор(ы)

6.52.2.1 Keybind()

```
Keybind.Keybind (
            string name,
            string key,
            string category )   [inline]
```

Конструктор класса

Аргументы

name	Название действия
key	Ключ действия
category	Категория действия

См. определение в файле DataBase.cs строка 302

```
00303     {
00304         this.name = name;
00305         this.key = key;
00306         this.category = category;
00307     }
```

6.52.3 Данные класса

6.52.3.1 category

string Keybind.category

См. определение в файле DataBase.cs строка 295

6.52.3.2 key

string Keybind.key

См. определение в файле DataBase.cs строка 294

6.52.3.3 name

string Keybind.name

См. определение в файле DataBase.cs строка 293

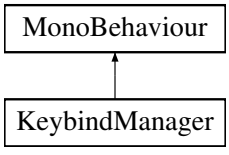
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/DataBase.cs

6.53 Класс KeybindManager

Модуль управления объектом настройки управления

Граф наследования:KeybindManager:



Открытые члены

- string GetName ()

  Получить название настройки управлеия
- KeyCode GetKey ()

  Получить заданный ключ настройки управления
- void SetName (string newName)

  Установить название настройки управления
- void SetKey (KeyCode newKey)

  Установить значение ключа настройки управления
- void SetKeybind ()

  Фунция для кпноки изменения значения ключа настройки управления
- void ResetKeybind ()

  Функция для обнуления значения ключа управления

Статические открытые данные

- static Action< KeybindManager, string, KeyCode > keyChanged
- static Action< bool > Binding

6.53.1 Подробное описание

Модуль управления объектом настройки управления

См. определение в файле KeybindManager.cs строка 10

6.53.2 Методы

6.53.2.1 GetKey()

KeyCode KeybindManager.GetKey ( ) [inline]

Получить заданный ключ настройки управления

Возвращает

Ключ настройки управления

См. определение в файле KeybindManager.cs строка 69

```
00070     {
00071         return key;
00072     }
```

6.53.2.2 GetName()

string KeybindManager.GetName ( ) [inline]

Получить название настройки управлеия

Возвращает

Название настройки

См. определение в файле KeybindManager.cs строка 61

```
00062     {
00063         return keybindName.text;
00064     }
```

6.53.2.3 ResetKeybind()

void KeybindManager.ResetKeybind ( ) [inline]

Функция для обнуления значения ключа управления

См. определение в файле KeybindManager.cs строка 102

```
00103     {
00104         keybindKeys.text = "None";
00105         key = KeyCode.None;
00106     }
```

6.53.2.4 SetKey()

void KeybindManager.SetKey (
KeyCode newKey) [inline]

Установить значение ключа настройки управления

Аргументы

newKey	Новый ключ настройки управления

См. определение в файле KeybindManager.cs строка 85

```
00086     {
00087         key = newKey;
00088         keybindKeys.text = KeyHandler.NormalizeKeybind(newKey);
00089     }
```

6.53.2.5 SetKeybind()

void KeybindManager.SetKeybind ( ) [inline]

Фунция для кпноки изменения значения ключа настройки управления

См. определение в файле KeybindManager.cs строка 93

```
00094     {
00095         binding = true;
00096         keybindKeys.text = "_";
00097         Binding?.Invoke(true);
00098     }
```

6.53.2.6 SetName()

void KeybindManager.SetName (
string newName) [inline]

Установить название настройки управления

Аргументы

newName	Новое название

См. определение в файле KeybindManager.cs строка 77

```
00078     {
00079         keybindName.text = newName;
00080     }
```

6.53.3 Данные класса

6.53.3.1 Binding

Action<bool> KeybindManager.Binding [static]

См. определение в файле KeybindManager.cs строка 16

6.53.3.2 keyChanged

Action<KeybindManager, string, KeyCode> KeybindManager.keyChanged [static]

См. определение в файле KeybindManager.cs строка 15

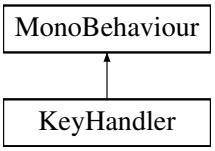
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/Managers/KeybindManager.cs

6.54 Класс KeyHandler

Модуль для всего, что связанно со считыванием нажатий на клавиатуру (привязка, считывание, регулирование)

Граф наследования:KeyHandler:



Открытые члены

- void Pause (bool pause)

  Регулирует "приостановку" игры. Во время паузы считывается только нажатие ESC

Аргументы

pause	

- bool IsPaused ()
- bool IsClearAndPlaying ()

  Индикатор того, что игра уже началась и не приостановлена, а также не открыты никакие окна UI
- void SetUIOpened (bool opened)

  Регулирует статус открытых меню (при нажатии ESC все меню сворачиваются, повтроное нажатие открывает меню паузы)
- bool GetUIOpened ()
- Dictionary< string, KeyCode > GetKeybinds ()
- KeyCode GetKeybind (string name)

  Привязанная к действию клавиша
- bool SetKeybind (string name, KeyCode key)

  -* Установить привязанную клавишу для действия

Открытые статические члены

- static string NormalizeKeybind (KeyCode code)

  Вспомогательная функция для преобразования названия ключа в более логичное

Статические открытые данные

- static KeyHandler instance
- static List< KeyCode > AllKeys
- static Action< string, KeyCode > keyPressed
- static string[] movementKeys
- static string[] abilityKeys
- static string[] uiKeys
- static KeyCode[] numbersKeyCodes

6.54.1 Подробное описание

Модуль для всего, что связанно со считыванием нажатий на клавиатуру (привязка, считывание, регулирование)

AllKeys - Список всех KeyCode

movementKeys - Список всех действий связанных с движением

abilityKeys - Список всех действий связанных с умениями персонажа

uiKeys - Список всех действий связанных с меню UI

keybinds - Все связанные действия и KeyCode

numbersKeyCodes - KeyCode для цифровых клавиш клавиатуры

См. определение в файле KeyHandler.cs строка 24

6.54.2 Методы

6.54.2.1 GetKeybind()

```
KeyCode KeyHandler.GetKeybind (
            string name )   [inline]
```

Привязанная к действию клавиша

Аргументы

name	Действие

Возвращает

Привязанная клавиша

См. определение в файле KeyHandler.cs строка 212

```
00213     {
00214         if (!keybinds.ContainsKey(name)) return KeyCode.None;
00215         return keybinds[name];
00216     }
```

6.54.2.2 GetKeybinds()

```
Dictionary< string, KeyCode > KeyHandler.GetKeybinds ( )   [inline]
```

Возвращает

Словарь действий и привязанных клавиш

См. определение в файле KeyHandler.cs строка 203

```
00204     {
00205         return keybinds;
00206     }
```

6.54.2.3 GetUIOpened()

bool KeyHandler.GetUIOpened ( ) [inline]

Возвращает

Открыто ли меню UI

См. определение в файле KeyHandler.cs строка 187

```
00188     {
00189         return uiOpened;
00190     }
```

6.54.2.4 IsClearAndPlaying()

bool KeyHandler.IsClearAndPlaying ( ) [inline]

Индикатор того, что игра уже началась и не приостановлена, а также не открыты никакие окна UI

Возвращает

См. определение в файле KeyHandler.cs строка 172

```
00173     {
00174         return !paused && !uiOpened && !spawnPause && !binding;
00175     }
```

6.54.2.5 IsPaused()

bool KeyHandler.IsPaused ( ) [inline]

Возвращает

Приостановлена ли игра

См. определение в файле KeyHandler.cs строка 164

```
00165     {
00166         return paused;
00167     }
```

6.54.2.6 NormalizeKeybind()

static string KeyHandler.NormalizeKeybind (
KeyCode code) [inline], [static]

Вспомогательная функция для преобразования названия ключа в более логичное

Аргументы

code	Ключ действия

Возвращает

Преобразованное название ключв

См. определение в файле KeyHandler.cs строка 238

```
00239     {
00240         string keyName = code.ToString();
00241         if (keyName.StartsWith("Alpha"))
00242         {
00243             return keyName[keyName.Length - 1].ToString();
00244         }
00245         else if (keyName.StartsWith("Mouse"))
00246         {
00247             switch (keyName)
00248             {
00249                 case "Mouse0":
00250                     return "LMB";
00251                 case "Mouse1":
00252                     return "RMB";
00253                 case "Mouse2":
00254                     return "MMB";
00255                 case "Mouse3":
00256                     return "SMB 1";
00257                 case "Mouse4":
00258                     return "SMB 2";
00259                 case "Mouse5":
00260                     return "SMB 3";
00261                 case "Mouse6":
00262                     return "SMB 4";
00263             }
00264         }
00265         return keyName;
00266     }
```

6.54.2.7 Pause()

void KeyHandler.Pause (
bool pause) [inline]

Регулирует "приостановку" игры. Во время паузы считывается только нажатие ESC

Аргументы

pause	

См. определение в файле KeyHandler.cs строка 155

```
00156     {
00157
00158         paused = pause;
00159     }
```

6.54.2.8 SetKeybind()

bool KeyHandler.SetKeybind (
string name,
KeyCode key) [inline]

-* Установить привязанную клавишу для действия

Аргументы

name	Действие
key	Клавиша

Возвращает

Была ли установлена указанная клавиша

См. определение в файле KeyHandler.cs строка 223

```
00224     {
00225         if (keybinds.ContainsValue(key) && keybinds[name] != key) // Исключить повторения
00226         {
00227             return false;
00228         }
00229         keybinds[name] = key;
00230         dataBase.SetKeybind(name, key);
00231         return true;
00232     }
```

6.54.2.9 SetUIOpened()

```
void KeyHandler.SetUIOpened (
                bool opened )   [inline]
```

Регулирует статус открытых меню (при нажатии ESC все меню сворачиваются, повтроное нажатие открывает меню паузы)

См. определение в файле KeyHandler.cs строка 179

```
00180     {
00181         uiOpened = opened;
00182     }
```

6.54.3 Данные класса

6.54.3.1 abilityKeys

```
string [] KeyHandler.abilityKeys   [static]
```

См. определение в файле KeyHandler.cs строка 37

6.54.3.2 AllKeys

```
List<KeyCode> KeyHandler.AllKeys   [static]
```

См. определение в файле KeyHandler.cs строка 31

6.54.3.3 instance

KeyHandler KeyHandler.instance [static]

См. определение в файле KeyHandler.cs строка 28

6.54.3.4 keyPressed

Action<string, KeyCode> KeyHandler.keyPressed [static]

См. определение в файле KeyHandler.cs строка 34

6.54.3.5 movementKeys

string [] KeyHandler.movementKeys [static]

См. определение в файле KeyHandler.cs строка 36

6.54.3.6 numbersKeyCodes

KeyCode [] KeyHandler.numbersKeyCodes [static]

Инициализатор
```
= {
        KeyCode.Alpha1,
        KeyCode.Alpha2,
        KeyCode.Alpha3,
        KeyCode.Alpha4,
        KeyCode.Alpha5,
        KeyCode.Alpha6,
        KeyCode.Alpha7,
        KeyCode.Alpha8,
        KeyCode.Alpha9,
    }
```

См. определение в файле KeyHandler.cs строка 45

6.54.3.7 uiKeys

string [] KeyHandler.uiKeys [static]

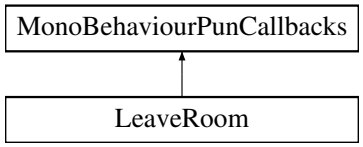
См. определение в файле KeyHandler.cs строка 38

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/KeyHandler.cs

6.55 Класс LeaveRoom

Граф наследования:LeaveRoom:



Открытые члены

- void LeaveFromRoom ()

  Метод для кнопки выйти из комнаты
- void ShowLeaveRoomButton ()
- void HideLeaveRoomButton ()

6.55.1 Подробное описание

См. определение в файле LeaveRoom.cs строка 5

6.55.2 Методы

6.55.2.1 HideLeaveRoomButton()

void LeaveRoom.HideLeaveRoomButton ( ) [inline]

См. определение в файле LeaveRoom.cs строка 24

```
00025     {
00026         leaveRoomButton.SetActive(false);
00027     }
```

6.55.2.2 LeaveFromRoom()

void LeaveRoom.LeaveFromRoom ( ) [inline]

Метод для кнопки выйти из комнаты

См. определение в файле LeaveRoom.cs строка 12

```
00013     {
00014         if (PhotonNetwork.InRoom)
00015         {
00016             PhotonNetwork.LeaveRoom();
00017         }
00018     }
```

6.55.2.3 ShowLeaveRoomButton()

void LeaveRoom.ShowLeaveRoomButton ( ) [inline]

См. определение в файле LeaveRoom.cs строка 20

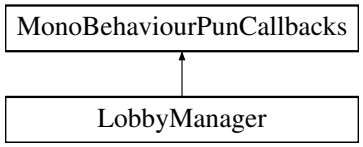
```
00021     {
00022         leaveRoomButton.SetActive(true);
00023     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Lobby/LeaveRoom.cs

6.56 Класс LobbyManager

Граф наследования:LobbyManager:



Открытые члены

- void CreateInviteFriend (int invitedUserID, string roomName=null)

  Метод служит для включения скрипа InviteFriend и заполнения его полей
- void CreateLobbyAndInviteUser (int invitedUserID)

  Метод который создает комнату и вызывает метод приглашения
- void AcceptInviteFriend (string roomName)

  Метод вызывает подключение к комнате фотон
- void ToggleSettings ()
- override void OnJoinedRoom ()
- override void OnConnectedToMaster ()

Открытые атрибуты

- string playerStatus

6.56.1 Подробное описание

См. определение в файле LobbyManager.cs строка 9

6.56.2 Методы

6.56.2.1 AcceptInviteFriend()

void LobbyManager.AcceptInviteFriend (
string roomName) [inline]

Метод вызывает подключение к комнате фотон

Аргументы

roomName	название комнаты

См. определение в файле LobbyManager.cs строка 87

```
00088     {
00089         var acceptFriendInvite = gameObject.GetComponent<AcceptFriendInvite>();
00090         acceptFriendInvite.enabled = true;
00091         acceptFriendInvite.roomName = roomName;
00092         acceptFriendInvite.StartAcceptInvite();
00093     }
```

6.56.2.2 CreateInviteFriend()

void LobbyManager.CreateInviteFriend (
int invitedUserID,
string roomName = null) [inline]

Метод служит для включения скрипа InviteFriend и заполнения его полей

Аргументы

invitedUserID	userID - приглашаемого
roomName	Имя комнаты куда нужно зайти приглашенному

См. определение в файле LobbyManager.cs строка 60

```
00061     {
00062         var inviteFriend = gameObject.GetComponent<InviteFriend>();
00063         inviteFriend.enabled = true;
00064         inviteFriend.invitedUserID = invitedUserID;
00065         inviteFriend.roomName = roomName;
00066         inviteFriend.StartInviteFriend();
00067     }
```

6.56.2.3 CreateLobbyAndInviteUser()

void LobbyManager.CreateLobbyAndInviteUser (
int invitedUserID) [inline]

Метод который создает комнату и вызывает метод приглашения

Аргументы

invitedUserID	userID приглашаемого

См. определение в файле LobbyManager.cs строка 73

```
00074     {
00075         var roomName = Random.Range(1000, 10000000).ToString();
00076         RoomOptions roomOptions = new RoomOptions() {IsVisible = false, PublishUserId = true};
00077         roomOptions.MaxPlayers = 2;
00078         CreateInviteFriend(invitedUserID);
00079         gameObject.GetComponent<LobbyManager>().playerStatus = "INLOBBY";
```

```
00080         PhotonNetwork.CreateRoom(roomName, roomOptions);
00081     }
```

6.56.2.4 OnConnectedToMaster()

override void LobbyManager.OnConnectedToMaster ( ) [inline]

См. определение в файле LobbyManager.cs строка 113

```
00114     {
00115         Debug.Log("YOU LEFT ROOM");
00116         gameObject.GetComponent<LeaveRoom>().HideLeaveRoomButton();
00117         gameObject.GetComponent<StartGameLevel>().enabled = false;
00118         gameObject.GetComponent<PlayersFounded>().HidePlayersFounded();
00119     }
```

6.56.2.5 OnJoinedRoom()

override void LobbyManager.OnJoinedRoom ( ) [inline]

См. определение в файле LobbyManager.cs строка 100

```
00101     {
00102         PhotonNetwork.LocalPlayer.NickName = username.text;
00103         if (playerStatus == "INLOBBY")
00104         {
00105             Debug.Log("YOU JOIN IN ROOM: " + PhotonNetwork.CurrentRoom.Name);
00106             gameObject.GetComponent<LeaveRoom>().ShowLeaveRoomButton();
00107         }else if (playerStatus == "SEARCHGAME")
00108         {
00109
00110         }
00111     }
```

6.56.2.6 ToggleSettings()

void LobbyManager.ToggleSettings ( ) [inline]

См. определение в файле LobbyManager.cs строка 95

```
00096     {
00097         settings.SetActive(!settings.activeSelf);
00098     }
```

6.56.3 Данные класса

6.56.3.1 playerStatus

string LobbyManager.playerStatus

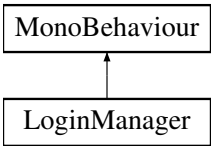
См. определение в файле LobbyManager.cs строка 17

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Lobby/LobbyManager.cs

6.57 Класс LoginManager

Граф наследования:LoginManager:



Открытые члены

- void LoginInAccount ()
  Авторизация в аккаунт
- void ShowRegisterScene ()

Открытые атрибуты

- InputField loginInput
- InputField passwordInput

6.57.1 Подробное описание

См. определение в файле LoginManager.cs строка 5

6.57.2 Методы

6.57.2.1 LoginInAccount()

void LoginManager.LoginInAccount ( ) [inline]

Авторизация в аккаунт

См. определение в файле LoginManager.cs строка 21

```
00022     {
00023         _controllers.Login(loginInput.text, passwordInput.text);
00024     }
```

6.57.2.2 ShowRegisterScene()

void LoginManager.ShowRegisterScene ( ) [inline]

См. определение в файле LoginManager.cs строка 26

```
00027     {
00028         SceneManager.LoadScene("Register");
00029     }
```

6.57.3 Данные класса

6.57.3.1 loginInput

InputField LoginManager.loginInput

См. определение в файле LoginManager.cs строка 9

6.57.3.2 passwordInput

InputField LoginManager.passwordInput

См. определение в файле LoginManager.cs строка 10

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Authorization/Login/LoginManager.cs

6.58 Класс StructsRequest.LoginStructRequest

Открытые атрибуты

- string login
- string password

6.58.1 Подробное описание

См. определение в файле Structs.cs строка 4

6.58.2 Данные класса

6.58.2.1 login

string StructsRequest.LoginStructRequest.login

См. определение в файле Structs.cs строка 6

6.58.2.2 password

string StructsRequest.LoginStructRequest.password

См. определение в файле Structs.cs строка 7

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Structs.cs

6.59 Класс ServersInfo.MainServerInfo

Статические открытые данные

- static string ServerDomain = "http://77.81.229.193:8000"

6.59.1 Подробное описание

См. определение в файле Structs.cs строка 69

6.59.2 Данные класса

6.59.2.1 ServerDomain

string ServersInfo.MainServerInfo.ServerDomain = "http://77.81.229.193:8000" [static]

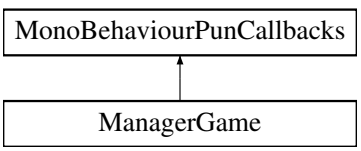
См. определение в файле Structs.cs строка 71

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Structs.cs

6.60 Класс ManagerGame

Граф наследования:ManagerGame:



Открытые члены

- override void OnPlayerPropertiesUpdate (Player targetPlayer, Hashtable changedProps)

  После того как игроки будут распределны по командам - заспавнить их

6.60.1 Подробное описание

См. определение в файле ManagerGame.cs строка 9

6.60.2 Методы

6.60.2.1 OnPlayerPropertiesUpdate()

```
override void ManagerGame.OnPlayerPropertiesUpdate (
                Player targetPlayer,
                Hashtable changedProps )   [inline]
```

После того как игроки будут распределны по командам - заспавнить их

Аргументы

targetPlayer	
changedProps	

См. определение в файле ManagerGame.cs строка 154

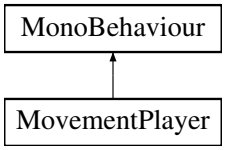
```
00155     {
00156         gameObject.GetComponent<SpawnEnemy>().enabled = true;
00157         //gameObject.GetComponent<SpawnPlayers>().enabled = true;
00158     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/ManagerGame.cs

6.61 Класс MovementPlayer

Граф наследования:MovementPlayer:



Открытые члены

- void cancelMovement (float stopTime)

Открытые атрибуты

- bool canMove = true

6.61.1 Подробное описание

См. определение в файле MovementPlayer.cs строка 5

6.61.2 Методы

6.61.2.1 cancelMovement()

```
void MovementPlayer.cancelMovement (
                float stopTime )   [inline]
```

См. определение в файле MovementPlayer.cs строка 36

```
00037     {
00038         StartCoroutine(movementStopCoroutine(stopTime));
00039     }
```

6.61.3 Данные класса

6.61.3.1 canMove

```
bool MovementPlayer.canMove = true
```

См. определение в файле MovementPlayer.cs строка 12

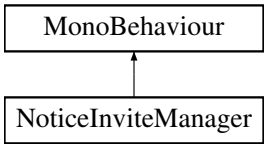
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Player/MovementPlayer.cs

6.62 Класс NoticeInviteManager

Модуль управления списком приглашений в группу (в лобби)

Граф наследования:NoticeInviteManager:



Открытые члены

- void AcceptInvite ()
- void CancelInvite ()

Открытые атрибуты

- Text usernameTextField
- string roomName
- LobbyManager lobbyManager

6.62.1 Подробное описание

Модуль управления списком приглашений в группу (в лобби)

См. определение в файле NoticeInviteManager.cs строка 8

6.62.2 Методы

6.62.2.1 AcceptInvite()

void NoticeInviteManager.AcceptInvite ( ) [inline]

См. определение в файле NoticeInviteManager.cs строка 14

```
00015     {
00016         lobbyManager.AcceptInviteFriend(roomName);
00017         Destroy(gameObject);
00018     }
```

6.62.2.2 CancelInvite()

void NoticeInviteManager.CancelInvite ( ) [inline]

См. определение в файле NoticeInviteManager.cs строка 20

```
00021     {
00022         Destroy(gameObject);
00023     }
```

6.62.3 Данные класса

6.62.3.1 lobbyManager

LobbyManager NoticeInviteManager.lobbyManager

См. определение в файле NoticeInviteManager.cs строка 12

6.62.3.2 roomName

string NoticeInviteManager.roomName

См. определение в файле NoticeInviteManager.cs строка 11

6.62.3.3 usernameTextField

Text NoticeInviteManager.usernameTextField

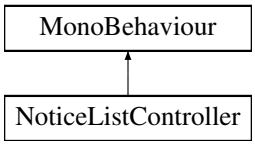
См. определение в файле NoticeInviteManager.cs строка 10

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/Managers/NoticeInviteManager.cs

6.63 Класс NoticeListController

Граф наследования:NoticeListController:



Открытые члены

- void OpenNoticeList ()
  Метод открывает/скрывает список уведомлений
- void AddInviteNotice (string inviteFromUserID, string roomName)
  Медот добавляет новое уведомление в ScrollView

6.63.1 Подробное описание

См. определение в файле NoticeListController.cs строка 3

6.63.2 Методы

6.63.2.1 AddInviteNotice()

void NoticeListController.AddInviteNotice (
string inviteFromUserID,
string roomName) [inline]

Медот добавляет новое уведомление в ScrollView

Аргументы

inviteFromUserID	userID игрока который пригласил
roomName	название команты, куда приглашают (Фотон комната)

См. определение в файле NoticeListController.cs строка 24

```
00025     {
00026         var noticeInvite = uiNoticeInvitePrefab.gameObject.GetComponent<NoticeInviteManager>();
00027         noticeInvite.usernameTextField.text = inviteFromUserID;
00028         noticeInvite.roomName = roomName;
00029         noticeInvite.lobbyManager = gameObject.GetComponent<LobbyManager>();
00030 
00031         var instance = Instantiate(noticeInvite.gameObject);
00032         instance.transform.SetParent(uiNoticeListScrollContent.transform, false);
00033     }
```

6.63.2.2 OpenNoticeList()

void NoticeListController.OpenNoticeList ( ) [inline]

Метод открывает/скрывает список уведомлений

См. определение в файле NoticeListController.cs строка 14

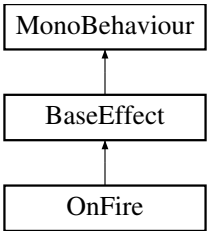
```
00015     {
00016         uiNoticeList.SetActive(!uiNoticeList.activeSelf);
00017     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Lobby/NoticeListController.cs

6.64 Класс OnFire

Граф наследования:OnFire:



Открытые члены

- override void ApplyEffect (GameObject entity)
- IEnumerator OnFireEffect (GameObject entity)

Дополнительные унаследованные члены

6.64.1 Подробное описание

См. определение в файле OnFire.cs строка 5

6.64.2 Методы

6.64.2.1 ApplyEffect()

override void OnFire.ApplyEffect (
GameObject entity) [inline], [virtual]

Переопределяет метод предка BaseEffect.

См. определение в файле OnFire.cs строка 11

```
00012     {
00013         base.ApplyEffect(entity);
00014         StartCoroutine(OnFireEffect(entity));
00015 
00016     }
```

6.64.2.2 OnFireEffect()

IEnumerator OnFire.OnFireEffect (
GameObject entity) [inline]

См. определение в файле OnFire.cs строка 18

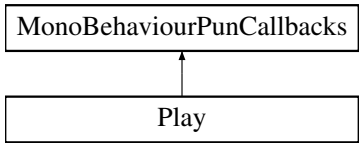
```
00018                                       {
00019         for(int i = 0; i < repeatCount; i++)
00020         {
00021             entity.GetComponent<BaseEntity>().TakeDamage(fireDamage);
00022             yield return new WaitForSeconds(damageTickSeconds);
00023         }
00024     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Effects/OnFire.cs

6.65 Класс Play

Граф наследования:Play:



Открытые члены

- void StartPlayInGame ()

  Событие кнопки Play
- override void OnFriendListUpdate (List< FriendInfo > friendsInfo)

  Обновление списка друзей, а именно, когда найден лидер в комнате присоединиться к нему
- override void OnConnectedToMaster ()

6.65.1 Подробное описание

См. определение в файле Play.cs строка 8

6.65.2 Методы

6.65.2.1 OnConnectedToMaster()

override void Play.OnConnectedToMaster ( ) [inline]

См. определение в файле Play.cs строка 153

```
00154     {
00155         if (gameObject.GetComponent<LobbyManager>().playerStatus == "SEARCHGAME")
00156         {
00157             GoInGame();
00158         }
00159 
00160     }
```

6.65.2.2 OnFriendListUpdate()

override void Play.OnFriendListUpdate (
List< FriendInfo > friendsInfo) [inline]

Обновление списка друзей, а именно, когда найден лидер в комнате присоединиться к нему

Аргументы

friendsInfo	

См. определение в файле Play.cs строка 138

```
00139     {
00140         foreach (var friend in friendsInfo)
00141         {
00142             if (friend.UserId == _masterClientIDGame)
00143             {
00144                 if (friend.IsInRoom)
00145                 {
00146                     _startMode = 4;
00147                     GoInGame(friend.Room);
00148                 }
00149             }
00150         }
```

```
00151     }
```

6.65.2.3 StartPlayInGame()

void Play.StartPlayInGame ( ) [inline]

Событие кнопки Play

См. определение в файле Play.cs строка 23

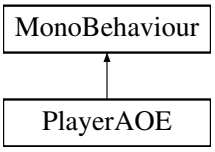
```
00024     {
00025         gameObject.GetComponent<LobbyManager>().playerStatus = "SEARCHGAME";
00026         // играет один
00027         if ((PhotonNetwork.InRoom && PhotonNetwork.PlayerList.Length == 1) || (PhotonNetwork.PlayerList.Length ==
      0))
00028         {
00029             PlayInSolo();
00030         }
00031 
00032         // играет не один (запускает только лидер комнаты)
00033         if (PhotonNetwork.PlayerList.Length > 1 && PhotonNetwork.IsMasterClient)
00034         {
00035             PlayInTeam();
00036         }
00037     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Lobby/Play.cs

6.66 Класс PlayerAOE

Граф наследования:PlayerAOE:



Классы

- struct AOEstruct

Открытые члены

- void Attack (string AOEKey)

6.66.1 Подробное описание

См. определение в файле PlayerAOE.cs строка 7

6.66.2 Методы

6.66.2.1 Attack()

```
void PlayerAOE.Attack (
                string AOEKey )   [inline]
```

См. определение в файле PlayerAOE.cs строка 36

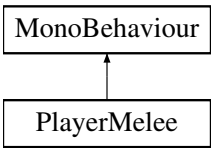
```
00037     {
00038         _view.RPC("CreateAOE", RpcTarget.All, AOEKey);
00039     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Attack/PlayerAOE.cs

6.67 Класс PlayerMelee

Граф наследования:PlayerMelee:



Открытые члены

- void MasterCheckMeleeAttack ()

6.67.1 Подробное описание

См. определение в файле PlayerMelee.cs строка 8

6.67.2 Методы

6.67.2.1 MasterCheckMeleeAttack()

void PlayerMelee.MasterCheckMeleeAttack ( ) [inline]

См. определение в файле PlayerMelee.cs строка 47

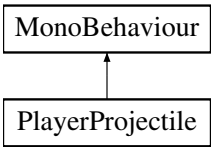
```
00048     {
00049         Debug.Log("Mellee check");
00050         if(_isTimeout) return;
00051         StartCoroutine(AttackTiemout());
00052 
00053         Collider2D[] hitEnemies = Physics2D.OverlapCircleAll(_attackPoint, attackRange, enemyLayers);
00054 
00055         foreach (Collider2D enemy in hitEnemies)
00056         {
00057             if (CanDamageThisEnemy(enemy))
00058             {
00059                 int dmg = damage + gameObject.GetComponent<BaseEntity>().GetBaseDamage();
00060                 _myView.RPC("TakeDamageRemote", RpcTarget.All,
        enemy.GetComponentInParent<PhotonView>().ViewID, dmg);
00061             }
00062         }
00063     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Attack/PlayerMelee.cs

6.68 Класс PlayerProjectile

Граф наследования:PlayerProjectile:



Классы

- struct Projectile

Открытые члены

- void Attack (string projectileKey)
- void Attack (string projectileKey, float time)

6.68.1 Подробное описание

См. определение в файле PlayerProjectile.cs строка 7

6.68.2 Методы

6.68.2.1 Attack() [1/2]

```
void PlayerProjectile.Attack (
                string projectileKey )   [inline]
```

См. определение в файле PlayerProjectile.cs строка 45

```
00046     {
00047         //RemoteProjectileAttack(_aimAngle, _aimDirection, projectileKey);
00048         _view.RPC("RemoteProjectileAttack", RpcTarget.All, _aimAngle, _aimDirection, projectileKey);
00049     }
```

6.68.2.2 Attack() [2/2]

```
void PlayerProjectile.Attack (
                string projectileKey,
                float time )   [inline]
```

См. определение в файле PlayerProjectile.cs строка 51

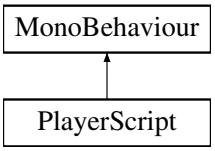
```
00052     {
00053         //RemoteProjectileAttack(_aimAngle, _aimDirection, projectileKey);
00054         if(!isTimeout){
00055             StartCoroutine(AttackTimeout(time));
00056             _view.RPC("RemoteProjectileAttack", RpcTarget.All, _aimAngle, _aimDirection, projectileKey);
00057         }
00058     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Attack/PlayerProjectile.cs

6.69 Класс PlayerScript

Граф наследования:PlayerScript:



Открытые члены

- void KillPlayer ()

6.69.1 Подробное описание

См. определение в файле PlayerScript.cs строка 5

6.69.2 Методы

6.69.2.1 KillPlayer()

void PlayerScript.KillPlayer ( ) [inline]

См. определение в файле PlayerScript.cs строка 59

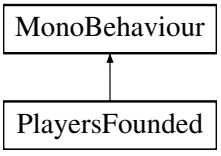
```
00060     {
00061         var myPlayerView = gameObject.GetComponent<PhotonView>();
00062 
00063         Debug.Log("PLAYER DIED FROM: " + myPlayerView.Owner.GetPhotonTeam().Name);
00064         if (myPlayerView.Owner.GetPhotonTeam().Name == "TeamOne")
00065         {
00066             _playersTeamsManager.PlayerInTeamOneDied();
00067         }
00068 
00069         if (myPlayerView.Owner.GetPhotonTeam().Name == "TeamTwo")
00070         {
00071             _playersTeamsManager.PlayerInTeamTwoDied();
00072         }
00073         gameObject.SetActive(false);
00074         //PhotonNetwork.Destroy(gameObject);
00075     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Player/PlayerScript.cs

6.70 Класс PlayersFounded

Граф наследования:PlayersFounded:



Открытые члены

- void UpdatePlayersFounded ()

  Обновляет текустовую информацию о текущем кол-ве игроков
- void ShowPlayersFounded ()

  Включает объект "PlayersFounded"
- void HidePlayersFounded ()

  Выключает объект "PlayersFounded"

6.70.1 Подробное описание

См. определение в файле PlayersFounded.cs строка 5

6.70.2 Методы

6.70.2.1 HidePlayersFounded()

void PlayersFounded.HidePlayersFounded ( ) [inline]

Выключает объект "PlayersFounded"

См. определение в файле PlayersFounded.cs строка 32

```
00033     {
00034         UpdatePlayersFounded();
00035         playersFounded.SetActive(false);
00036         stopSearchGameButton.SetActive(false);
00037         Debug.Log("HIDE here" + Time.deltaTime);
00038     }
```

6.70.2.2 ShowPlayersFounded()

void PlayersFounded.ShowPlayersFounded ( ) [inline]

Включает объект "PlayersFounded"

См. определение в файле PlayersFounded.cs строка 22

```
00023     {
00024         playersFounded.SetActive(true);
00025         stopSearchGameButton.SetActive(true);
00026         Debug.Log("SHOW here" + Time.deltaTime);
00027     }
```

6.70.2.3 UpdatePlayersFounded()

void PlayersFounded.UpdatePlayersFounded ( ) [inline]

Обновляет текустовую информацию о текущем кол-ве игроков

См. определение в файле PlayersFounded.cs строка 14

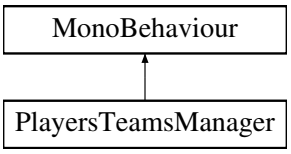
```
00015     {
00016         playersFoundedText.text = PhotonNetwork.PlayerList.Length + " / " + GameSettingsOriginal.MaxPlayersInGame
      + " founded";
00017     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Lobby/PlayersFounded.cs

6.71 Класс PlayersTeamsManager

Граф наследования:PlayersTeamsManager:



Открытые члены

- void PlayerInTeamOneDied ()
- void PlayerInTeamTwoDied ()

6.71.1 Подробное описание

См. определение в файле PlayersTeamsManager.cs строка 5

6.71.2 Методы

6.71.2.1 PlayerInTeamOneDied()

void PlayersTeamsManager.PlayerInTeamOneDied ( ) [inline]

См. определение в файле PlayersTeamsManager.cs строка 17

```
00018     {
00019         _lifePlayersInTeamOne -= 1;
00020         if (_lifePlayersInTeamOne == 0)
00021         {
00022             _endGame.TeamOneWin();
00023         }
00024     }
```

6.71.2.2 PlayerInTeamTwoDied()

void PlayersTeamsManager.PlayerInTeamTwoDied ( ) [inline]

См. определение в файле PlayersTeamsManager.cs строка 26

```
00027     {
00028         _lifePlayersInTeamTwo -= 1;
00029         if (_lifePlayersInTeamTwo == 0)
00030         {
00031             _endGame.TeamTwoWin();
00032         }
00033     }
```

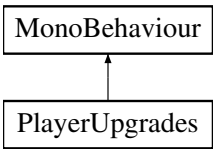
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Teams/PlayersTeamsManager.cs

6.72 Класс PlayerUpgrades

Модуль управления панелью улучшения игрока !НЕ АКТИВНО!

Граф наследования:PlayerUpgrades:



Открытые члены

- int GetXp ()
- void AddXp (int points)
- void AddUpgrade (Button btn)
- void SwitchPanels (bool toClass=true)

6.72.1 Подробное описание

Модуль управления панелью улучшения игрока !НЕ АКТИВНО!

См. определение в файле PlayerUpgrades.cs строка 10

6.72.2 Методы

6.72.2.1 AddUpgrade()

```
void PlayerUpgrades.AddUpgrade (
                Button btn )   [inline]
```

См. определение в файле PlayerUpgrades.cs строка 62

```
00063     {
00064         if (xpPoints >= btn.GetComponent<BaseUpgrade>().GetCost())
00065         {
00066             xpPoints -= btn.GetComponent<BaseUpgrade>().GetCost();
00067             btn.onClick.RemoveAllListeners();
00068             Color col = btn.GetComponent<Image>().color;
00069             col.a = 0.1f;
00070             btn.GetComponent<BaseUpgrade>().ApplyUpgrade(player);
00071         }
00072 
00073     }
```

6.72.2.2 AddXp()

```
void PlayerUpgrades.AddXp (
                int points )   [inline]
```

См. определение в файле PlayerUpgrades.cs строка 57

```
00058     {
00059         if (points < 0) return;
00060         xpPoints += points;
00061     }
```

6.72.2.3 GetXp()

```
int PlayerUpgrades.GetXp ( )   [inline]
```

См. определение в файле PlayerUpgrades.cs строка 53

```
00054     {
00055         return xpPoints;
00056     }
```

6.72.2.4 SwitchPanels()

```
void PlayerUpgrades.SwitchPanels (
                bool toClass = true )   [inline]
```

См. определение в файле PlayerUpgrades.cs строка 74

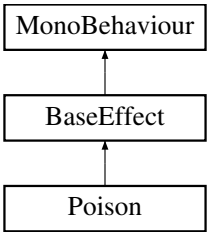
```
00075     {
00076         classPanel.gameObject.SetActive(toClass);
00077         subclassPanel.gameObject.SetActive(!toClass);
00078     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Player/PlayerUpgrades.cs

6.73 Класс Poison

Граф наследования:Poison:



Открытые члены

- override void ApplyEffect (GameObject entity)
- IEnumerator PoisonEffect (GameObject entity)

Открытые атрибуты

- int posionDamage = 5
- float damageTickSeconds = 1.0f

6.73.1 Подробное описание

См. определение в файле Poison.cs строка 6

6.73.2 Методы

6.73.2.1 ApplyEffect()

override void Poison.ApplyEffect (
GameObject entity) [inline], [virtual]

Переопределяет метод предка BaseEffect.

См. определение в файле Poison.cs строка 11

```
00012     {
00013         base.ApplyEffect(entity);
00014         entity.gameObject.GetComponent<MonoBehaviour>().StartCoroutine(PoisonEffect(entity));
00015     }
```

6.73.2.2 PoisonEffect()

IEnumerator Poison.PoisonEffect (
GameObject entity) [inline]

См. определение в файле Poison.cs строка 17

```
00017                                                 {
00018         for(int i = 0; i < duration/damageTickSeconds; i++)
00019         {
00020             Debug.Log("Poisoned " + entity.name);
00021             entity.GetComponent<PhotonView>().RPC("TakeDamageRemote", RpcTarget.All,
        entity.GetComponent<PhotonView>().ViewID, posionDamage);
00022             yield return new WaitForSeconds(damageTickSeconds);
00023         }
00024     }
```

6.73.3 Данные класса

6.73.3.1 damageTickSeconds

float Poison.damageTickSeconds = 1.0f

См. определение в файле Poison.cs строка 9

6.73.3.2 posionDamage

int Poison.posionDamage = 5

См. определение в файле Poison.cs строка 8

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Effects/Poison.cs

6.74 Структура PlayerProjectile.Projectile

Открытые атрибуты

- string projectileKey
- GameObject projectile

6.74.1 Подробное описание

См. определение в файле PlayerProjectile.cs строка 17

6.74.2 Данные класса

6.74.2.1 projectile

GameObject PlayerProjectile.Projectile.projectile

См. определение в файле PlayerProjectile.cs строка 20

6.74.2.2 projectileKey

string PlayerProjectile.Projectile.projectileKey

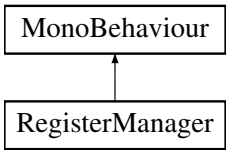
См. определение в файле PlayerProjectile.cs строка 19

Объявления и описания членов структуры находятся в файле:

- Vuji/Assets/Scripts/Game/Attack/PlayerProjectile.cs

6.75 Класс RegisterManager

Граф наследования:RegisterManager:



Открытые члены

- void RegisterAccount ()

  Регистрация аккаунта + проверка корректности заполненных полей
- void ShowLoginScene ()

Открытые атрибуты

- InputField loginInput
- InputField passwordOneInput
- InputField passwordTwoInput

6.75.1 Подробное описание

См. определение в файле RegisterManager.cs строка 7

6.75.2 Методы

6.75.2.1 RegisterAccount()

void RegisterManager.RegisterAccount ( ) [inline]

Регистрация аккаунта + проверка корректности заполненных полей

См. определение в файле RegisterManager.cs строка 23

```
00024     {
00025         if (passwordOneInput.text != passwordTwoInput.text)
00026         {
00027             Debug.Log("password error");
00028             return;
00029         }
00030         _controllers.Register(loginInput.text, passwordOneInput.text);
00031     }
```

6.75.2.2 ShowLoginScene()

void RegisterManager.ShowLoginScene ( ) [inline]

См. определение в файле RegisterManager.cs строка 33

```
00034     {
00035         SceneManager.LoadScene("Login");
00036     }
```

6.75.3 Данные класса

6.75.3.1 loginInput

InputField RegisterManager.loginInput

См. определение в файле RegisterManager.cs строка 10

6.75.3.2 passwordOneInput

InputField RegisterManager.passwordOneInput

См. определение в файле RegisterManager.cs строка 11

6.75.3.3 passwordTwoInput

InputField RegisterManager.passwordTwoInput

См. определение в файле RegisterManager.cs строка 12

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Authorization/Register/RegisterManager.cs

6.76 Класс StructsRequest.RegisterStructRequest

Открытые атрибуты

- string login
- string password

6.76.1 Подробное описание

См. определение в файле Structs.cs строка 11

6.76.2 Данные класса

6.76.2.1 login

string StructsRequest.RegisterStructRequest.login

См. определение в файле Structs.cs строка 13

6.76.2.2 password

string StructsRequest.RegisterStructRequest.password

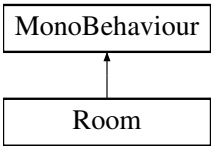
См. определение в файле Structs.cs строка 14

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Structs.cs

6.77 Класс Room

Граф наследования:Room:



Открытые члены

- void RotateRandomly ()

6.77.1 Подробное описание

См. определение в файле Room.cs строка 5

6.77.2 Методы

6.77.2.1 RotateRandomly()

```
void Room.RotateRandomly ( )   [inline]
```

См. определение в файле Room.cs строка 13

```
00014     {
00015         int count = Random.Range(0, 4);
00016 
00017         for (int i = 0; i < count; i++)
00018         {
00019             transform.Rotate(0, 90, 0);
00020 
00021         }
00022     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Map/Room.cs

6.78 Класс Setting

Вспомогательный класс для передачи сохраненных настроек

Открытые члены

- Setting (string name, string value)
  Конструктор класса

Открытые атрибуты

- string name
- string value

6.78.1 Подробное описание

Вспомогательный класс для передачи сохраненных настроек

См. определение в файле DataBase.cs строка 312

6.78.2 Конструктор(ы)

6.78.2.1 Setting()

```
Setting.Setting (
            string name,
            string value )   [inline]
```

Конструктор класса

Аргументы

name	Название
value	Значение

См. определение в файле DataBase.cs строка 321

```
00322     {
00323         this.name = name;
00324         this.value = value;
00325     }
```

6.78.3 Данные класса

6.78.3.1 name

string Setting.name

См. определение в файле DataBase.cs строка 314

6.78.3.2 value

string Setting.value

См. определение в файле DataBase.cs строка 315

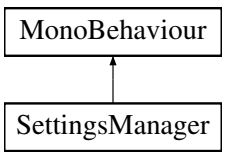
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/DataBase.cs

6.79 Класс SettingsManager

Модуль управления настройками

Граф наследования:SettingsManager:



Открытые члены

- void OnKeybindMovementToggle (Toggle keybindMvmToggle)

  Фукнция для переключения движения по заданным кнопка (или заданным в движке юнити)
- void ChangeFullscreen (Toggle fullscreenToggle)

  Функция для переключателя режима полного экрана окна игры в настройках видео
- void ChangeResolution (Dropdown dropdown)

  Фунция для выпадающего списка установки размеров окна игры в настройках видео
- void ChangeFps (Slider fpsSlider)

  Функция для слайдера целевых ФПС в настройках видео
- void ChangeVsync (Toggle vsyncToggle)

  Фукнция для переключателя Вертикальной синхронизации в настройках видео
- void SwitchPanels (GameObject toPanel)

  Функция переключения панелей настроек

Статические открытые данные

- static Action< bool > keybindMovementToggled

6.79.1 Подробное описание

Модуль управления настройками

См. определение в файле SettingsManager.cs строка 10

6.79.2 Методы

6.79.2.1 ChangeFps()

void SettingsManager.ChangeFps (
Slider fpsSlider) [inline]

Функция для слайдера целевых ФПС в настройках видео

Аргументы

fpsSlider	Целевой слайдер

См. определение в файле SettingsManager.cs строка 184

```
00185     {
00186         Application.targetFrameRate = int.Parse(fpsSlider.value.ToString());
00187         dataBase.SetSetting("FPS", fpsSlider.value.ToString());
00188         maxFpsText.text = fpsSlider.value.ToString();
00189     }
```

6.79.2.2 ChangeFullscreen()

void SettingsManager.ChangeFullscreen (
Toggle fullscreenToggle) [inline]

Функция для переключателя режима полного экрана окна игры в настройках видео

Аргументы

fullscreenToggle	Целевой переключатель

См. определение в файле SettingsManager.cs строка 159

```
00160     {
00161         fullscreen = fullscreenToggle.isOn;
00162         dataBase.SetSetting("Fullscreen", fullscreen.ToString());
00163         SetScreenResolution();
00164     }
```

6.79.2.3 ChangeResolution()

void SettingsManager.ChangeResolution (
Dropdown dropdown) [inline]

Фунция для выпадающего списка установки размеров окна игры в настройках видео

Аргументы

dropdown	Целевой выпадающий список

См. определение в файле SettingsManager.cs строка 170

```
00171     {
00172         dropdown.Hide();
00173         string[] resolution = dropdown.captionText.text.ToString().Split('x'); ;
00174         Debug.Log(string.Join(" ", resolution));
00175         int.TryParse(resolution[0], out resolutionX);
00176         int.TryParse(resolution[1], out resolutionY);
00177         dataBase.SetSetting("Resolution", dropdown.value.ToString());
00178         SetScreenResolution();
00179     }
```

6.79.2.4 ChangeVsync()

void SettingsManager.ChangeVsync (
Toggle vsyncToggle) [inline]

Фукнция для переключателя Вертикальной синхронизации в настройках видео

Аргументы

vsyncToggle	Переключатель настройки

См. определение в файле SettingsManager.cs строка 194
```
00195     {
00196         QualitySettings.vSyncCount = Convert.ToInt32(vsyncToggle.isOn);
00197         dataBase.SetSetting("VSync", vsyncToggle.isOn.ToString());
00198     }
```

6.79.2.5 OnKeybindMovementToggle()

void SettingsManager.OnKeybindMovementToggle (
Toggle keybindMvmToggle) [inline]

Фукнция для переключения движения по заданным кнопка (или заданным в движке юнити)

Аргументы

keybindMvmToggle	

См. определение в файле SettingsManager.cs строка 125
```
00126     {
00127         movementBlocker.gameObject.SetActive(!keybindMvmToggle.isOn);
00128         keybindMovementToggled?.Invoke(keybindMvmToggle.isOn);
00129     }
```

6.79.2.6 SwitchPanels()

void SettingsManager.SwitchPanels (
GameObject toPanel) [inline]

Функция переключения панелей настроек

Аргументы

toPanel	Панель, на которую необходимо переключиться

См. определение в файле SettingsManager.cs строка 213
```
00214     {
00215         videoSettingsPanel.SetActive(false);
00216         midSettingsPanel.SetActive(false);
00217         keybindsPanel.SetActive(false);
00218         toPanel.SetActive(true);
00219     }
```

6.79.3 Данные класса

6.79.3.1 keybindMovementToggled

Action<bool> SettingsManager.keybindMovementToggled [static]

См. определение в файле SettingsManager.cs строка 13

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/Managers/SettingsManager.cs

6.80 Структура BaseEntity.Skill

Открытые атрибуты

- string key
- GameObject skill

6.80.1 Подробное описание

См. определение в файле BaseEntity.cs строка 27

6.80.2 Данные класса

6.80.2.1 key

string BaseEntity.Skill.key

См. определение в файле BaseEntity.cs строка 29

6.80.2.2 skill

GameObject BaseEntity.Skill.skill

См. определение в файле BaseEntity.cs строка 30

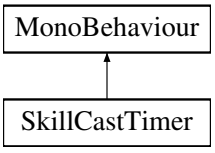
Объявления и описания членов структуры находятся в файле:

- Vuji/Assets/Scripts/Game/BaseObjects/BaseEntity.cs

6.81 Класс SkillCastTimer

Модуль для управления таймером каста скила целевой сущностью

Граф наследования:SkillCastTimer:



Открытые члены

- void StartTimer (float time)

  Функция для события каста скила (Запустить таймер каста с указанным временем)

6.81.1 Подробное описание

Модуль для управления таймером каста скила целевой сущностью

См. определение в файле SkillCastTimer.cs строка 9

6.81.2 Методы

6.81.2.1 StartTimer()

```
void SkillCastTimer.StartTimer (
            float time )  [inline]
```

Функция для события каста скила (Запустить таймер каста с указанным временем)

Аргументы

time	Время каста умения

См. определение в файле SkillCastTimer.cs строка 29

```
00030     {
00031         timerValue = 0f;
00032         timerInterval = time;
00033         timerSlider.minValue = 0f;
00034         timerSlider.maxValue = time;
00035     }
```

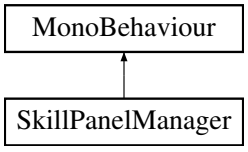
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/Units/SkillCastTimer.cs

6.82 Класс SkillPanelManager

Модуль для управления панелью скилов сущности

Граф наследования:SkillPanelManager:



Открытые атрибуты

- GameObject trackedPlayer

6.82.1 Подробное описание

Модуль для управления панелью скилов сущности

См. определение в файле SkillPanelManager.cs строка 7

6.82.2 Данные класса

6.82.2.1 trackedPlayer

GameObject SkillPanelManager.trackedPlayer

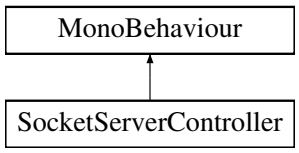
См. определение в файле SkillPanelManager.cs строка 11

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/Managers/SkillPanelManager.cs

6.83 Класс SocketServerController

Граф наследования:SocketServerController:



Открытые члены

- void StartSendInviteToSocketServer (int invitedUserID, string roomName)
  открытие потока на отправку приглашения другому игроку
- void CloseConnection ()
  Метод отключается от SocketServer

6.83.1 Подробное описание

См. определение в файле SocketServerController.cs строка 8

6.83.2 Методы

6.83.2.1 CloseConnection()

void SocketServerController.CloseConnection ( ) [inline]

Метод отключается от SocketServer

См. определение в файле SocketServerController.cs строка 168

```
00169     {
00170         _tcpSocket.Shutdown(SocketShutdown.Both);
00171         _tcpSocket.Close();
00172     }
```

6.83.2.2 StartSendInviteToSocketServer()

void SocketServerController.StartSendInviteToSocketServer (
int invitedUserID,
string roomName) [inline]

открытие потока на отправку приглашения другому игроку

Аргументы

invitedUserID	id приглашаемого пользователя
roomName	название комнаты куда должен присоединиться приглашенный

См. определение в файле SocketServerController.cs строка 159

```
00160     {
00161         _thread = new Thread(() => { SendInviteToSocketServer(invitedUserID, roomName); });
00162         _thread.Start();
00163     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Lobby/SocketServerController.cs

6.84 Класс ServersInfo.SocketServerInfo

Статические открытые данные

- static string SocketServerIP = "77.81.229.193"
- static int SocketServerPort = 5000
- static string CommandOpenConnect = "600"
- static string CommandInvite = "700"
- static string CommandHaveInvite = "701"
- static string CommandOpenConnectServer = "600S"
- static string CommandInviteServer = "700S"
- static string CommandHaveInviteServer = "701S"

6.84.1 Подробное описание

См. определение в файле Structs.cs строка 75

6.84.2 Данные класса

6.84.2.1 CommandHaveInvite

string ServersInfo.SocketServerInfo.CommandHaveInvite = "701" [static]

См. определение в файле Structs.cs строка 83

6.84.2.2 CommandHaveInviteServer

string ServersInfo.SocketServerInfo.CommandHaveInviteServer = "701S" [static]

См. определение в файле Structs.cs строка 88

6.84.2.3 CommandInvite

string ServersInfo.SocketServerInfo.CommandInvite = "700" [static]

См. определение в файле Structs.cs строка 80

6.84.2.4 CommandInviteServer

string ServersInfo.SocketServerInfo.CommandInviteServer = "700S" [static]

См. определение в файле Structs.cs строка 87

6.84.2.5 CommandOpenConnect

string ServersInfo.SocketServerInfo.CommandOpenConnect = "600" [static]

См. определение в файле Structs.cs строка 79

6.84.2.6 CommandOpenConnectServer

string ServersInfo.SocketServerInfo.CommandOpenConnectServer = "600S" [static]

См. определение в файле Structs.cs строка 86

6.84.2.7 SocketServerIP

string ServersInfo.SocketServerInfo.SocketServerIP = "77.81.229.193" [static]

См. определение в файле Structs.cs строка 77

6.84.2.8 SocketServerPort

int ServersInfo.SocketServerInfo.SocketServerPort = 5000 [static]

См. определение в файле Structs.cs строка 78

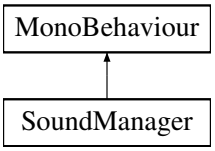
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Structs.cs

6.85 Класс SoundManager

Модуль для уравления звуком целевого источника

Граф наследования:SoundManager:



6.85.1 Подробное описание

Модуль для уравления звуком целевого источника

См. определение в файле SoundManager.cs строка 7

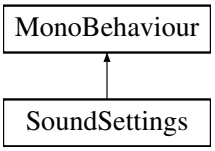
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Sounds/SoundManager.cs

6.86 Класс SoundSettings

Модуль для управления настройками звука

Граф наследования:SoundSettings:



Открытые члены

- void SoundSliderValueChange (string name, float value)

  Функция для события установки значения звука пользователем
- float GetVolume (string name)

  Получить значение звука для заданной категории

Статические открытые данные

- static Action< string, float > volumeChange
- static SoundSettings instance

6.86.1 Подробное описание

Модуль для управления настройками звука

См. определение в файле SoundSettings.cs строка 9

6.86.2 Методы

6.86.2.1 GetVolume()

float SoundSettings.GetVolume (
string name) [inline]

Получить значение звука для заданной категории

Аргументы

name	Название категории

Возвращает

Громкость категории

См. определение в файле SoundSettings.cs строка 61

```
00062     {
00063         return volumeList[name];
00064     }
```

6.86.2.2 SoundSliderValueChange()

```
void SoundSettings.SoundSliderValueChange (
                string name,
                float value )   [inline]
```

Функция для события установки значения звука пользователем

Аргументы

name	
value	

См. определение в файле SoundSettings.cs строка 50

```
00051     {
00052         volumeList[name] = value;
00053         dataBase.SetSetting(name, value.ToString());
00054         volumeChange?.Invoke(name, value);
00055     }
```

6.86.3 Данные класса

6.86.3.1 instance

```
SoundSettings SoundSettings.instance   [static]
```

См. определение в файле SoundSettings.cs строка 15

6.86.3.2 volumeChange

```
Action<string, float> SoundSettings.volumeChange   [static]
```

См. определение в файле SoundSettings.cs строка 14

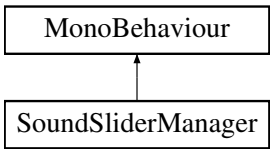
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/Managers/SoundSettings.cs

6.87 Класс SoundSliderManager

Модуль управления слайдером для настройки звука

Граф наследования:SoundSliderManager:



Открытые члены

- void ValueChanged ()

  Функция для привязки к изменению значения слайдера
- void SetName (string newName)

  Установить название настройки звука
- void SetValue (float value)

  Установить значение настройки звука

Статические открытые данные

- static Action< string, float > onValueChange

6.87.1 Подробное описание

Модуль управления слайдером для настройки звука

См. определение в файле SoundSliderManager.cs строка 9

6.87.2 Методы

6.87.2.1 SetName()

void SoundSliderManager.SetName (
string newName) [inline]

Установить название настройки звука

Аргументы

newName	Новое название

См. определение в файле SoundSliderManager.cs строка 34
```
00035     {
00036         text.text = newName;
00037     }
```

6.87.2.2 SetValue()

void SoundSliderManager.SetValue (
float value) [inline]

Установить значение настройки звука

Аргументы

value	Новое значение

См. определение в файле SoundSliderManager.cs строка 42
```
00043     {
00044         oldValue = value;
00045         slider.value = value;
00046     }
```

6.87.2.3 ValueChanged()

void SoundSliderManager.ValueChanged ( ) [inline]

Функция для привязки к изменению значения слайдера

См. определение в файле SoundSliderManager.cs строка 20
```
00021     {
00022         if (oldValue != slider.value)
00023         {
00024             onValueChange?.Invoke(text.text, slider.value);
00025             oldValue = slider.value;
00026         }
00027
00028     }
```

6.87.3 Данные класса

6.87.3.1 onValueChange

Action<string, float> SoundSliderManager.onValueChange [static]

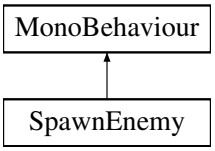
См. определение в файле SoundSliderManager.cs строка 14

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/Managers/SoundSliderManager.cs

6.88 Класс SpawnEnemy

Граф наследования:SpawnEnemy:



Открытые атрибуты

- GameObject goblinSeekerGameObject2

6.88.1 Подробное описание

См. определение в файле SpawnEnemy.cs строка 5

6.88.2 Данные класса

6.88.2.1 goblinSeekerGameObject2

GameObject SpawnEnemy.goblinSeekerGameObject2

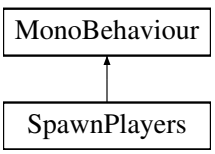
См. определение в файле SpawnEnemy.cs строка 7

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Spawn/SpawnEnemy.cs

6.89 Класс SpawnPlayers

Граф наследования:SpawnPlayers:



Открытые члены

- void SetPlayerObject (GameObject prefab)

Статические открытые данные

- static Action< GameObject > OnSpawn

6.89.1 Подробное описание

См. определение в файле SpawnPlayers.cs строка 6

6.89.2 Методы

6.89.2.1 SetPlayerObject()

```
void SpawnPlayers.SetPlayerObject (
                GameObject prefab )   [inline]
```

См. определение в файле SpawnPlayers.cs строка 45

```
00046     {
00047         playerGameObject = prefab;
00048     }
```

6.89.3 Данные класса

6.89.3.1 OnSpawn

```
Action<GameObject> SpawnPlayers.OnSpawn   [static]
```

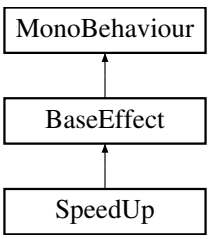
См. определение в файле SpawnPlayers.cs строка 10

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Spawn/SpawnPlayers.cs

6.90 Класс SpeedUp

Граф наследования:SpeedUp:



Открытые члены

- override void ApplyEffect (GameObject entity)
- IEnumerator SpeedEffect (GameObject entity)

Открытые атрибуты

- int additionalSpeed = 2

6.90.1 Подробное описание

См. определение в файле SpeedUp.cs строка 5

6.90.2 Методы

6.90.2.1 ApplyEffect()

override void SpeedUp.ApplyEffect (
GameObject entity) [inline], [virtual]

Переопределяет метод предка BaseEffect.

См. определение в файле SpeedUp.cs строка 10

```
00011     {
00012         base.ApplyEffect(entity);
00013         entity.gameObject.GetComponent<MonoBehaviour>().StartCoroutine(SpeedEffect(entity));
00014     }
```

6.90.2.2 SpeedEffect()

IEnumerator SpeedUp.SpeedEffect (
GameObject entity) [inline]

См. определение в файле SpeedUp.cs строка 16

```
00016                                           {
00017         entity.GetComponent<BaseEntity>().IncreaseSpeed(additionalSpeed);
00018         yield return new WaitForSeconds(duration);
00019         entity.GetComponent<BaseEntity>().DecreaseSpeed(additionalSpeed);
00020     }
```

6.90.3 Данные класса

6.90.3.1 additionalSpeed

int SpeedUp.additionalSpeed = 2

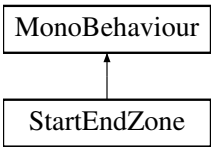
См. определение в файле SpeedUp.cs строка 8

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Effects/SpeedUp.cs

6.91 Класс StartEndZone

Граф наследования:StartEndZone:



6.91.1 Подробное описание

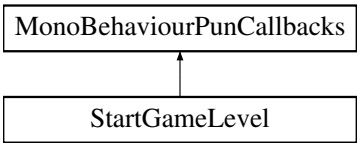
См. определение в файле StartEndZone.cs строка 5

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Map/StartEndZone.cs

6.92 Класс StartGameLevel

Граф наследования:StartGameLevel:



Открытые члены

- override void OnJoinedRoom ()
- override void OnPlayerEnteredRoom (Player newPlayer)
- override void OnPlayerLeftRoom (Player newPlayer)

6.92.1 Подробное описание

См. определение в файле StartGameLevel.cs строка 8

6.92.2 Методы

6.92.2.1 OnJoinedRoom()

override void StartGameLevel.OnJoinedRoom ( ) [inline]

См. определение в файле StartGameLevel.cs строка 24

```
00025     {
00026         if (gameObject.GetComponent<LobbyManager>().playerStatus == "SEARCHGAME")
00027         {
00028             _playersFounded.ShowPlayersFounded();
00029             if (CheckPreGameRoom())
00030             {
00031                 LoadGameLevel();
00032             }
00033 
00034             _playersFounded.UpdatePlayersFounded();
00035         }
00036     }
```

6.92.2.2 OnPlayerEnteredRoom()

override void StartGameLevel.OnPlayerEnteredRoom (
Player newPlayer) [inline]

См. определение в файле StartGameLevel.cs строка 38

```
00039     {
00040         if (gameObject.GetComponent<LobbyManager>().playerStatus == "SEARCHGAME")
00041         {
00042             if (CheckPreGameRoom())
00043             {
00044                 LoadGameLevel();
00045             }
00046 
00047             _playersFounded.UpdatePlayersFounded();
00048         }
00049     }
```

6.92.2.3 OnPlayerLeftRoom()

override void StartGameLevel.OnPlayerLeftRoom (
Player newPlayer) [inline]

См. определение в файле StartGameLevel.cs строка 51

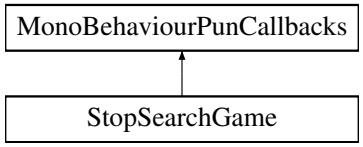
```
00052     {
00053         _playersFounded.UpdatePlayersFounded();
00054     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Lobby/StartGameLevel.cs

6.93 Класс StopSearchGame

Граф наследования:StopSearchGame:



Открытые члены

- void CancelSearchGame ()
- override void OnConnectedToMaster ()
- override void OnFriendListUpdate (List< FriendInfo > friendsInfo)

6.93.1 Подробное описание

См. определение в файле StopSearchGame.cs строка 6

6.93.2 Методы

6.93.2.1 CancelSearchGame()

void StopSearchGame.CancelSearchGame ( ) [inline]

См. определение в файле StopSearchGame.cs строка 12

```
00013     {
00014         var myTeamID = PhotonNetwork.LocalPlayer.CustomProperties["team"].ToString();
00015         if (myTeamID != "None")
00016         {
00017             foreach (var player in PhotonNetwork.PlayerListOthers)
00018             {
00019                 var playerTeamID = player.CustomProperties["team"].ToString();
00020                 if (myTeamID == playerTeamID)
00021                 {
00022                     _playerInTeam.Add(player.UserId);
00023                     var view = PhotonView.Get(this);
00024                     view.RPC("LeaveFromSearch", RpcTarget.Others, player.UserId, PhotonNetwork.LocalPlayer.UserId);
00025                     _startMode = 2;
00026                     gameObject.GetComponent<LobbyManager>().playerStatus = "INLOBBY";
00027                     PhotonNetwork.LeaveRoom();
00028                 }
00029             }
00030         }
00031         else
00032         {
00033             gameObject.GetComponent<LobbyManager>().playerStatus = "INLOBBY";
00034             PhotonNetwork.LeaveRoom();
00035         }
00036     }
```

6.93.2.2 OnConnectedToMaster()

override void StopSearchGame.OnConnectedToMaster ( ) [inline]

См. определение в файле StopSearchGame.cs строка 55

```
00056     {
00057         JoinToFriendTeam();
00058     }
```

6.93.2.3 OnFriendListUpdate()

override void StopSearchGame.OnFriendListUpdate (
List< FriendInfo > friendsInfo) [inline]

См. определение в файле StopSearchGame.cs строка 60

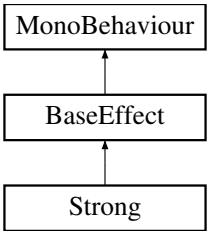
```
00061     {
00062         foreach (var friend in friendsInfo)
00063         {
00064             if (friend.UserId == _teammateID)
00065             {
00066                 if (friend.IsInRoom)
00067                 {
00068                     _startMode = 4;
00069                     JoinToFriendTeam(friend.Room);
00070                 }
00071             }
00072         }
00073     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Lobby/StopSearchGame.cs

6.94 Класс Strong

Граф наследования:Strong:



Открытые члены

- override void ApplyEffect (GameObject entity)
- IEnumerator StrongEffect (GameObject entity)

Открытые атрибуты

- int additionalDamage = 30

6.94.1 Подробное описание

См. определение в файле Strong.cs строка 5

6.94.2 Методы

6.94.2.1 ApplyEffect()

override void Strong.ApplyEffect (
GameObject entity) [inline], [virtual]

Переопределяет метод предка BaseEffect.

См. определение в файле Strong.cs строка 10

```
00011     {
00012         base.ApplyEffect(entity);
00013         entity.gameObject.GetComponent<MonoBehaviour>().StartCoroutine(StrongEffect(entity));
00014     }
```

6.94.2.2 StrongEffect()

IEnumerator Strong.StrongEffect (
GameObject entity) [inline]

См. определение в файле Strong.cs строка 16

```
00016                                           {
00017         entity.GetComponent<BaseEntity>().IncreaseDamage(additionalDamage);
00018         yield return new WaitForSeconds(duration);
00019         entity.GetComponent<BaseEntity>().DecreaseDamage(additionalDamage);
00020     }
```

6.94.3 Данные класса

6.94.3.1 additionalDamage

int Strong.additionalDamage = 30

См. определение в файле Strong.cs строка 8

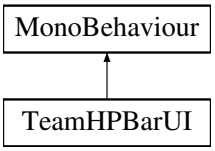
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Effects/Strong.cs

6.95 Класс TeamHPBarUI

Модуль для управления отдельным хп баром в панели списка участников команды

Граф наследования:TeamHPBarUI:



Открытые члены

- void SetEntity (BaseEntity player, string name)
  Функция установки целевой сущности

6.95.1 Подробное описание

Модуль для управления отдельным хп баром в панели списка участников команды

См. определение в файле TeamHPBarUI.cs строка 8

6.95.2 Методы

6.95.2.1 SetEntity()

```
void TeamHPBarUI.SetEntity (
            BaseEntity player,
            string name )   [inline]
```

Функция установки целевой сущности

Аргументы

player	Целевая сущность
name	Имя целевого игрока

См. определение в файле TeamHPBarUI.cs строка 43

```
00044     {
00045         entity = player;
00046         HealthBarText.text = name;
00047     }
```

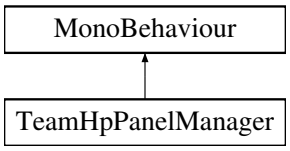
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/Units/TeamHPBarUI.cs

6.96 Класс TeamHpPanelManager

Модуль управления панелью списка участников команды

Граф наследования:TeamHpPanelManager:



6.96.1 Подробное описание

Модуль управления панелью списка участников команды

См. определение в файле TeamHpPanelManager.cs строка 7

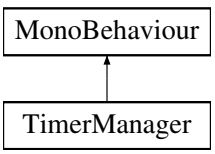
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/Managers/TeamHpPanelManager.cs

6.97 Класс TimerManager

Модуль управления таймером выбора класса

Граф наследования:TimerManager:



Статические открытые данные

- static Action< bool > timerEnd

6.97.1 Подробное описание

Модуль управления таймером выбора класса

См. определение в файле TimerManager.cs строка 9

6.97.2 Данные класса

6.97.2.1 timerEnd

Action<bool> TimerManager.timerEnd [static]

См. определение в файле TimerManager.cs строка 11

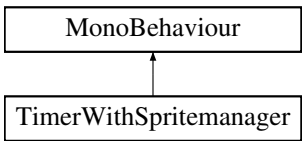
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/Managers/TimerManager.cs

6.98 Класс TimerWithSpritemanager

Модуль для управления отображаемым объектом скила

Граф наследования:TimerWithSpritemanager:



Открытые члены

- void SetTime (float time)
- void SetEntity (GameObject player, BaseSkill skill, string keyName)
  Установить сущность, скил и ключ для отслеживания

6.98.1 Подробное описание

Модуль для управления отображаемым объектом скила

См. определение в файле TimerWithSpritemanager.cs строка 9

6.98.2 Методы

6.98.2.1 SetEntity()

```
void TimerWithSpritemanager.SetEntity (
            GameObject player,
            BaseSkill skill,
            string keyName )  [inline]
```

Установить сущность, скил и ключ для отслеживания

Аргументы

player	Целевая сущность
skill	Целевой скилл
keyName	Название ключа действия дял скила

См. определение в файле TimerWithSpritemanager.cs строка 42

```
00043     {
00044         targetPlayer = player;
00045         targetSkill = skill;
00046         this.keyName = keyName;
00047         targetedSprite.sprite = skill.GetSprite();
00048         keyText.text = KeyHandler.NormalizeKeybind(KeyHandler.instance.GetKeybind(keyName));
00049         targetedText.text = "";
00050         skill.onRelease += SetTime;
00051         tooltipText.text = "\"" + skill.GetName() + "\"\n" + skill.GetDescription() + "\n" + "Cast time: " +
      skill.GetCastTime().ToString() + "s\n" + "Energy cost: " + skill.GetCost().ToString() + "\n" + "Cooldown: " +
      skill.GetCooldownTime().ToString() + "s";
00052         player.GetComponent<BaseEntity>().OnSkillSelectionChange += SetSelection;
00053     }
```

6.98.2.2 SetTime()

```
void TimerWithSpritemanager.SetTime (
                float time )   [inline]
```

См. определение в файле TimerWithSpritemanager.cs строка 26

```
00027     {
00028         timerInterval = time;
00029         timerValue = 0f;
00030     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/Managers/TimerWithSpritemanager.cs

6.99 Класс StructsResponse.TokenStructResponse

Открытые атрибуты

- string token

6.99.1 Подробное описание

См. определение в файле Structs.cs строка 33

6.99.2 Данные класса

6.99.2.1 token

string StructsResponse.TokenStructResponse.token

См. определение в файле Structs.cs строка 35

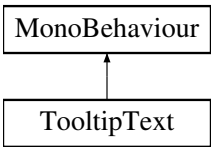
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Structs.cs

6.100 Класс TooltipText

Модуль для игровых объектов, при наведении на которых будет показана подсказка

Граф наследования:TooltipText:



Открытые атрибуты

- string text

6.100.1 Подробное описание

Модуль для игровых объектов, при наведении на которых будет показана подсказка

См. определение в файле TooltipText.cs строка 6

6.100.2 Данные класса

6.100.2.1 text

string TooltipText.text

См. определение в файле TooltipText.cs строка 8

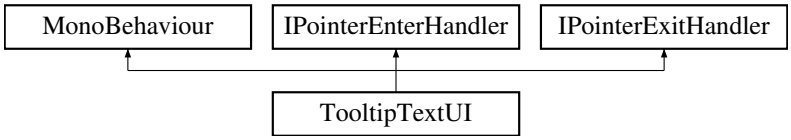
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/Units/TooltipText.cs

6.101 Класс TooltipTextUI

Модуль для объектов UI, при наведении на которых будет показана подсказка

Граф наследования:TooltipTextUI:



Открытые атрибуты

- string text

6.101.1 Подробное описание

Модуль для объектов UI, при наведении на которых будет показана подсказка

См. определение в файле TooltipTextUI.cs строка 7

6.101.2 Данные класса

6.101.2.1 text

string TooltipTextUI.text

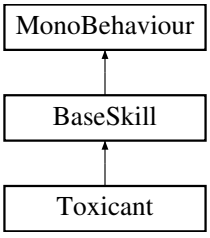
См. определение в файле TooltipTextUI.cs строка 10

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/Units/TooltipTextUI.cs

6.102 Класс Toxicant

Граф наследования:Toxicant:



Открытые члены

- override IEnumerator UseSkill (GameObject caster, string key)

Дополнительные унаследованные члены

6.102.1 Подробное описание

См. определение в файле Toxicant.cs строка 6

6.102.2 Методы

6.102.2.1 UseSkill()

```
override IEnumerator Toxicant.UseSkill (
                GameObject caster,
                string key )   [inline], [virtual]
```

Переопределяет метод предка BaseSkill.

См. определение в файле Toxicant.cs строка 8

```
00009     {
00010         base.UseSkill(caster, key);
00011         if (caster.GetComponent<BaseEntity>().spendEnergy(energyCost) == false)
00012         {
00013             Debug.Log("Not enough energy to cast " + this.gameObject.name);
00014             yield break;
00015         }
00016 
00017         onCast?.Invoke(castTime);
00018         caster.GetComponent<BaseEntity>().setIsCooldown(key, true);
00019         if(cancelMovementOnCast)
00020             caster.GetComponent<MovementPlayer>().cancelMovement(castTime);
00021         yield return new WaitForSeconds(castTime);
00022 
00023         // Сам скилл
00024         caster.GetComponent<PlayerAOE>().Attack("AcidFloor");
00025         // Сам скилл
00026         onRelease?.Invoke(cooldown);
00027         yield return new WaitForSeconds(cooldown);
00028         caster.GetComponent<BaseEntity>().setIsCooldown(key, false);
00029     }
```

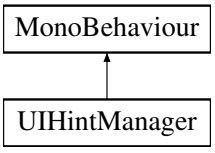
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Game/Skills/Toxicant.cs

6.103 Класс UIHintManager

Вспомогательный модуль для отображения подсказок на экране. Для отображения подсказки на объекте добавьте скрипт TooltipText (Для Ui - TooltipTextUI)

Граф наследования:UIHintManager:



Открытые типы

- enum ProjectMode { Tooltip3D = 0 , Tooltip2D = 1 }

Открытые атрибуты

- Color BGColor = Color.white
- Color textColor = Color.black
- ProjectMode tooltipMode = ProjectMode.Tooltip3D
- int fontSize = 14
- int maxWidth = 250
- int border = 10
- RectTransform box
- RectTransform arrow
- Text boxText
- Camera _camera
- float speed = 10

Статические открытые данные

- static string text
- static bool isUI

6.103.1 Подробное описание

Вспомогательный модуль для отображения подсказок на экране. Для отображения подсказки на объекте добавьте скрипт TooltipText (Для Ui - TooltipTextUI)

ВНИМАНИЕ: Код спизжен!

См. определение в файле UIHintManager.cs строка 12

6.103.2 Перечисления

6.103.2.1 ProjectMode

enum UIHintManager.ProjectMode

См. определение в файле UIHintManager.cs строка 20
00020 { Tooltip3D = 0, Tooltip2D = 1 };

6.103.3 Данные класса

6.103.3.1 _camera

Camera UIHintManager._camera

См. определение в файле UIHintManager.cs строка 28

6.103.3.2 arrow

RectTransform UIHintManager.arrow

См. определение в файле UIHintManager.cs строка 26

6.103.3.3 BGColor

Color UIHintManager.BGColor = Color.white

См. определение в файле UIHintManager.cs строка 18

6.103.3.4 border

int UIHintManager.border = 10

См. определение в файле UIHintManager.cs строка 24

6.103.3.5 box

RectTransform UIHintManager.box

См. определение в файле UIHintManager.cs строка 25

6.103.3.6 boxText

Text UIHintManager.boxText

См. определение в файле UIHintManager.cs строка 27

6.103.3.7 fontSize

int UIHintManager.fontSize = 14

См. определение в файле UIHintManager.cs строка 22

6.103.3.8 isUI

bool UIHintManager.isUI [static]

См. определение в файле UIHintManager.cs строка 16

6.103.3.9 maxWidth

int UIHintManager.maxWidth = 250

См. определение в файле UIHintManager.cs строка 23

6.103.3.10 speed

float UIHintManager.speed = 10

См. определение в файле UIHintManager.cs строка 29

6.103.3.11 text

string UIHintManager.text [static]

См. определение в файле UIHintManager.cs строка 15

6.103.3.12 textColor

Color UIHintManager.textColor = Color.black

См. определение в файле UIHintManager.cs строка 19

6.103.3.13 tooltipMode

ProjectMode UIHintManager.tooltipMode = ProjectMode.Tooltip3D

См. определение в файле UIHintManager.cs строка 21

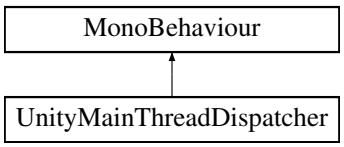
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UIScripts/Managers/UIHintManager.cs

6.104 Класс UnityMainThreadDispatcher

A thread-safe class which holds a queue with actions to execute on the next Update() method. It can be used to make calls to the main thread for things such as UI Manipulation in Unity. It was developed for use in combination with the Firebase Unity plugin, which uses separate threads for event handling

Граф наследования:UnityMainThreadDispatcher:



Открытые члены

- void Update ()
- void Enqueue (IEnumerator action)

  Locks the queue and adds the IEnumerator to the queue
- void Enqueue (Action action)

  Locks the queue and adds the Action to the queue
- Task EnqueueAsync (Action action)

  Locks the queue and adds the Action to the queue, returning a Task which is completed when the action completes

Открытые статические члены

- static bool Exists ()
- static UnityMainThreadDispatcher Instance ()

6.104.1 Подробное описание

A thread-safe class which holds a queue with actions to execute on the next Update() method. It can be used to make calls to the main thread for things such as UI Manipulation in Unity. It was developed for use in combination with the Firebase Unity plugin, which uses separate threads for event handling

Author: Pim de Witte (pimdewitte.com) and contributors, https://github.com/PimDeWitte/Unity↩MainThreadDispatcher

См. определение в файле UnityMainThreadDispatcher.cs строка 28

6.104.2 Методы

6.104.2.1 Enqueue() [1/2]

```
void UnityMainThreadDispatcher.Enqueue (
                Action action )   [inline]
```

Locks the queue and adds the Action to the queue

Аргументы

action	function that will be executed from the main thread.

См. определение в файле UnityMainThreadDispatcher.cs строка 56

```
00057     {
00058         Enqueue(ActionWrapper(action));
00059     }
```

6.104.2.2 Enqueue() [2/2]

```
void UnityMainThreadDispatcher.Enqueue (
                IEnumerator action )   [inline]
```

Locks the queue and adds the IEnumerator to the queue

Аргументы

action	IEnumerator function that will be executed from the main thread.

См. определение в файле UnityMainThreadDispatcher.cs строка 44

```
00044                                     {
00045         lock (_executionQueue) {
00046             _executionQueue.Enqueue (() => {
00047                 StartCoroutine (action);
00048             });
00049         }
00050     }
```

6.104.2.3 EnqueueAsync()

```
Task UnityMainThreadDispatcher.EnqueueAsync (
                Action action )   [inline]
```

Locks the queue and adds the Action to the queue, returning a Task which is completed when the action completes

Аргументы

action	function that will be executed from the main thread.

Возвращает

A Task that can be awaited until the action completes

См. определение в файле UnityMainThreadDispatcher.cs строка 66

```
00067     {
00068         var tcs = new TaskCompletionSource<bool>();
00069 
00070         void WrappedAction() {
00071             try
00072             {
00073                 action();
00074                 tcs.TrySetResult(true);
00075             } catch (Exception ex)
00076             {
00077                 tcs.TrySetException(ex);
00078             }
00079         }
00080 
00081         Enqueue(ActionWrapper(WrappedAction));
00082         return tcs.Task;
00083     }
```

6.104.2.4 Exists()

static bool UnityMainThreadDispatcher.Exists ( ) [inline], [static]

См. определение в файле UnityMainThreadDispatcher.cs строка 95

```
00095                           {
00096         return _instance != null;
00097     }
```

6.104.2.5 Instance()

static UnityMainThreadDispatcher UnityMainThreadDispatcher.Instance ( ) [inline], [static]

См. определение в файле UnityMainThreadDispatcher.cs строка 99

```
00099                                           {
00100         if (!Exists ()) {
00101             throw new Exception ("UnityMainThreadDispatcher could not find the UnityMainThreadDispatcher object.
      Please ensure you have added the MainThreadExecutor Prefab to your scene.");
00102         }
00103         return _instance;
00104     }
```

6.104.2.6 Update()

void UnityMainThreadDispatcher.Update ( ) [inline]

См. определение в файле UnityMainThreadDispatcher.cs строка 32

```
00032                 {
00033         lock(_executionQueue) {
00034             while (_executionQueue.Count > 0) {
00035                 _executionQueue.Dequeue().Invoke();
00036             }
00037         }
00038     }
```

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UnityMainThreadDispatcher.cs

6.105 Класс StructsResponse.UserIDStructResponse

Открытые атрибуты

- string userID

6.105.1 Подробное описание

См. определение в файле Structs.cs строка 39

6.105.2 Данные класса

6.105.2.1 userID

string StructsResponse.UserIDStructResponse.userID

См. определение в файле Structs.cs строка 41

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Structs.cs

6.106 Класс StructsResponse.UserInfoObject

Открытые атрибуты

- int userID
- string username

6.106.1 Подробное описание

См. определение в файле Structs.cs строка 59

6.106.2 Данные класса

6.106.2.1 userID

int StructsResponse.UserInfoObject.userID

См. определение в файле Structs.cs строка 61

6.106.2.2 username

string StructsResponse.UserInfoObject.username

См. определение в файле Structs.cs строка 62

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Structs.cs

6.107 Класс StructsResponse.UserInfoStructResponse

Открытые атрибуты

- string login
- string username
- string created_at

6.107.1 Подробное описание

См. определение в файле Structs.cs строка 45

6.107.2 Данные класса

6.107.2.1 created_at

string StructsResponse.UserInfoStructResponse.created_at

См. определение в файле Structs.cs строка 49

6.107.2.2 login

string StructsResponse.UserInfoStructResponse.login

См. определение в файле Structs.cs строка 47

6.107.2.3 username

string StructsResponse.UserInfoStructResponse.username

См. определение в файле Structs.cs строка 48

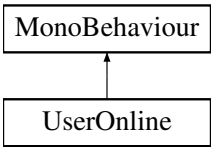
Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Structs.cs

6.108 Класс UserOnline

Вспомогательный модуль для сообщения серверу о том, что пользователь онлайн

Граф наследования:UserOnline:



6.108.1 Подробное описание

Вспомогательный модуль для сообщения серверу о том, что пользователь онлайн

См. определение в файле UserOnline.cs строка 5

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/UserOnline.cs

6.109 Класс StructsRequest.UserOnlineAndOfflineStructRequest

Открытые атрибуты

- string data

6.109.1 Подробное описание

См. определение в файле Structs.cs строка 18

6.109.2 Данные класса

6.109.2.1 data

string StructsRequest.UserOnlineAndOfflineStructRequest.data

См. определение в файле Structs.cs строка 20

Объявления и описания членов класса находятся в файле:

- Vuji/Assets/Scripts/Structs.cs

Глава 7

Файлы

7.1 LoginManager.cs

```
00001 using UnityEngine;
00002 using UnityEngine.UI;
00003 using UnityEngine.SceneManagement;
00004
00005 public class LoginManager : MonoBehaviour
00006 {
00007     // я определяю объекты в Unity, но можно и конструкции Find, GetComponent
00008     private Controllers _controllers;
00009     public InputField loginInput;
00010     public InputField passwordInput;
00011
00012     void Start()
00013     {
00014         Screen.SetResolution(1054, 593, false);
00015         _controllers = GetComponent<Controllers>();
00016     }
00017
00021     public void LoginInAccount()
00022     {
00023         _controllers.Login(loginInput.text, passwordInput.text);
00024     }
00025
00026     public void ShowRegisterScene()
00027     {
00028         SceneManager.LoadScene("Register");
00029     }
00030 }
```

7.2 RegisterManager.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 using UnityEngine.UI;
00005 using UnityEngine.SceneManagement;
00006
00007 public class RegisterManager : MonoBehaviour
00008 {
00009     private Controllers _controllers;
00010     public InputField loginInput;
00011     public InputField passwordOneInput;
00012     public InputField passwordTwoInput;
00013
00014     void Start()
00015     {
00016         Screen.SetResolution(1054, 593, false);
00017         _controllers = GetComponent<Controllers>();
00018     }
00019
00023     public void RegisterAccount()
00024     {
00025         if (passwordOneInput.text != passwordTwoInput.text)
00026         {
00027             Debug.Log("password error");
```

```
00028             return;
00029         }
00030         _controllers.Register(loginInput.text, passwordOneInput.text);
00031     }
00032
00033     public void ShowLoginScene()
00034     {
00035         SceneManager.LoadScene("Login");
00036     }
00037 }
```

7.3 Controllers.cs

```
00001 using System.Collections;
00002 using StructsRequest;
00003 using StructsResponse;
00004 using ServersInfo;
00005 using UnityEngine;
00006 using UnityEngine.SceneManagement;
00007 using UnityEngine.Networking;
00008 using UnityEngine.UI;
00009
00013 public class Controllers : MonoBehaviour
00014 {
00015     private readonly string _serverDomain = MainServerInfo.ServerDomain; // Адрес сервера
00016     private DataBase _dataBase; // База данных из объекта на котором висит скрипт
00017
00018     #region Unity Methods
00019
00020     private void Start()
00021     {
00022         _dataBase = gameObject.GetComponent<DataBase>();
00023     }
00024
00025     #endregion
00026
00027     #region Public Methods
00031     public void CheckVujiServer()
00032     {
00033         StartCoroutine(CheckVujiServerNet());
00034     }
00040     public void Login(string login, string password)
00041     {
00042         StartCoroutine(LoginNet(login, password));
00043     }
00049     public void Register(string login, string password)
00050     {
00051         StartCoroutine(RegisterNet(login, password));
00052     }
00056     public void UserOnline()
00057     {
00058         string token = _dataBase.GetToken();
00059         StartCoroutine(UserOnlineNet(token));
00060     }
00064     public void UserOffline()
00065     {
00066         string token = _dataBase.GetToken();
00067         _dataBase.SetToken("");
00068         StartCoroutine(UserOfflineNet(token));
00069     }
00074     public void GetUserID(string token)
00075     {
00076         StartCoroutine(GetUserIDNet(token));
00077     }
00082     public void FindFriendsByName(string friendsName)
00083     {
00084         string token = _dataBase.GetToken();
00085         StartCoroutine(FindFriendsByNameNet(token, friendsName));
00086     }
00091     public void SetLocalUserName(Text field)
00092     {
00093         if (_dataBase == null) _dataBase = gameObject.GetComponent<DataBase>(); ;
00094         string token = _dataBase.GetToken();
00095         StartCoroutine(GetUserInfo(token, field));
00096
00097
00098     }
00099
00100     #endregion
00101
00102     #region Private IEnumerator Methods
00106     private void AutoAuth()
00107     {
00108         StartCoroutine(AutoAuthNet());
```

```
00109     }
00113     private IEnumerator AutoAuthNet()
00114     {
00115         WWWForm form = new WWWForm();
00116         string token = _dataBase.GetToken();
00117 
00118         UnityWebRequest www = UnityWebRequest.Post(_serverDomain + "/auth", form);
00119         www.SetRequestHeader("Authorization", token);
00120         yield return www.SendWebRequest();
00121         if (www.result == UnityWebRequest.Result.ProtocolError)
00122         {
00123             SceneManager.LoadScene("Login");
00124         }
00125         else
00126         {
00127             TokenStructResponse tokenStructResponse =
00128                 JsonUtility.FromJson<TokenStructResponse>(www.downloadHandler.text);
00129             _dataBase.SetToken(tokenStructResponse.token);
00130             SceneManager.LoadScene("Lobby");
00131         }
00132     }
00133 
00134     private IEnumerator CheckVujiServerNet()
00135     {
00136         UnityWebRequest www = UnityWebRequest.Get(_serverDomain);
00137         yield return www.SendWebRequest();
00138         if (www.result != UnityWebRequest.Result.Success)
00139         {
00140             Debug.Log("Vuji server: [OFFLINE]");
00141         }
00142         else
00143         {
00144             Debug.Log("Vuji server: [ONLINE]");
00145             AutoAuth();
00146         }
00147     }
00148 
00149     private IEnumerator LoginNet(string login, string password)
00150     {
00151         WWWForm form = new WWWForm();
00152 
00153         LoginStructRequest loginStruct = new LoginStructRequest();
00154         loginStruct.login = login;
00155         loginStruct.password = password;
00156         string json = JsonUtility.ToJson(loginStruct);
00157         UnityWebRequest www = UnityWebRequest.Post(_serverDomain + "/login", form);
00158         byte[] postBytes = System.Text.Encoding.UTF8.GetBytes(json);
00159         UploadHandler uploadHandler = new UploadHandlerRaw(postBytes);
00160 
00161         www.uploadHandler = uploadHandler;
00162         www.SetRequestHeader("Content-Type", "application/json; charset=UTF-8");
00163 
00164 
00165         yield return www.SendWebRequest();
00166         if (www.result == UnityWebRequest.Result.ProtocolError)
00167         {
00168             Debug.Log("Incorrect fields");
00169         }
00170         else
00171         {
00172             TokenStructResponse tokenStructResponse =
00173                 JsonUtility.FromJson<TokenStructResponse>(www.downloadHandler.text);
00174             _dataBase.SetToken(tokenStructResponse.token);
00175             SceneManager.LoadScene("Lobby");
00176         }
00177     }
00178 
00179     private IEnumerator RegisterNet(string login, string password)
00180     {
00181         WWWForm form = new WWWForm();
00182 
00183         RegisterStructRequest registerStructRequest = new RegisterStructRequest();
00184         registerStructRequest.login = login;
00185         registerStructRequest.password = password;
00186 
00187         string json = JsonUtility.ToJson(registerStructRequest);
00188         UnityWebRequest www = UnityWebRequest.Post(_serverDomain + "/register", form);
00189         byte[] postBytes = System.Text.Encoding.UTF8.GetBytes(json);
00190         UploadHandler uploadHandler = new UploadHandlerRaw(postBytes);
00191 
00192         www.uploadHandler = uploadHandler;
00193         www.SetRequestHeader("Content-Type", "application/json; charset=UTF-8");
00194 
00195 
00196         yield return www.SendWebRequest();
00197         if (www.result == UnityWebRequest.Result.ProtocolError)
00198         {
```

```
00199           Debug.Log("Login already used");
00200       }
00201       else
00202       {
00203           TokenStructResponse tokenStructResponse =
00204               JsonUtility.FromJson<TokenStructResponse>(www.downloadHandler.text);
00205           _dataBase.SetToken(tokenStructResponse.token);
00206           SceneManager.LoadScene("Login");
00207       }
00208   }
00209
00210   private IEnumerator UserOnlineNet(string token)
00211   {
00212       UserOnlineAndOfflineStructRequest userStruct = new UserOnlineAndOfflineStructRequest();
00213       userStruct.data = "None";
00214       string json = JsonUtility.ToJson(userStruct);
00215       UnityWebRequest www = UnityWebRequest.Put(_serverDomain + "/user_online", json);
00216       www.SetRequestHeader("Authorization", token);
00217       www.SetRequestHeader("Content-Type", "application/json; charset=UTF-8");
00218       yield return www.SendWebRequest();
00219   }
00220
00221   private IEnumerator UserOfflineNet(string token)
00222   {
00223       UserOnlineAndOfflineStructRequest userStruct = new UserOnlineAndOfflineStructRequest();
00224       userStruct.data = "None";
00225       string json = JsonUtility.ToJson(userStruct);
00226       UnityWebRequest www = UnityWebRequest.Put(_serverDomain + "/user_offline", json);
00227       www.SetRequestHeader("Authorization", token);
00228       www.SetRequestHeader("Content-Type", "application/json; charset=UTF-8");
00229       yield return www.SendWebRequest();
00230   }
00231
00232   private IEnumerator GetUserIDNet(string token)
00233   {
00234       UnityWebRequest www = UnityWebRequest.Get(_serverDomain + "/user_id");
00235       www.SetRequestHeader("Authorization", token);
00236       yield return www.SendWebRequest();
00237       if (www.result == UnityWebRequest.Result.ProtocolError)
00238       {
00239           Debug.Log("ERROR: cant set user_id in Photon.NickMame");
00240       }
00241       else
00242       {
00243           UserIDStructResponse userIDStructResponse =
00244               JsonUtility.FromJson<UserIDStructResponse>(www.downloadHandler.text);
00245           string userID = userIDStructResponse.userID;
00246       }
00247   }
00248
00249   private IEnumerator FindFriendsByNameNet(string token, string friendsName)
00250   {
00251       WWWForm form = new WWWForm();
00252       FindFriendsByNameStructRequest findFriendsByNameStructRequest = new FindFriendsByNameStructRequest();
00253       findFriendsByNameStructRequest.friendsName = friendsName;
00254
00255       string json = JsonUtility.ToJson(findFriendsByNameStructRequest);
00256       UnityWebRequest www = UnityWebRequest.Post(_serverDomain + "/find_friends_by_name", form);
00257       byte[] postBytes = System.Text.Encoding.UTF8.GetBytes(json);
00258       UploadHandler uploadHandler = new UploadHandlerRaw(postBytes);
00259
00260       www.uploadHandler = uploadHandler;
00261       www.SetRequestHeader("Content-Type", "application/json; charset=UTF-8");
00262       www.SetRequestHeader("Authorization", token);
00263       yield return www.SendWebRequest();
00264       if (www.result == UnityWebRequest.Result.ProtocolError)
00265       {
00266           Debug.Log("Unexpected error findFriends");
00267       }
00268       else
00269       {
00270           FindFriendsByNameStructResponse findFriendsByNameStructResponse =
00271               JsonUtility.FromJson<FindFriendsByNameStructResponse>("{\"friends\":" + www.downloadHandler.text +
    "}");
00272           UserInfoObject[] userInfoObject = findFriendsByNameStructResponse.friends;
00273
00274           FriendsListController friendsListController = gameObject.GetComponent<FriendsListController>();
00275           friendsListController.FillFriendsList(userInfoObject);
00276       }
00277   }
00278
00279   private IEnumerator GetUserInfo(string token, Text field = null)
00280   {
00281       UserOnlineAndOfflineStructRequest userStruct = new UserOnlineAndOfflineStructRequest();
00282       userStruct.data = "None";
00283       string json = JsonUtility.ToJson(userStruct);
00284       UnityWebRequest www = UnityWebRequest.Get(_serverDomain + "/me");
```

```
00285         www.SetRequestHeader("Authorization", token);
00286         www.SetRequestHeader("Content-Type", "application/json; charset=UTF-8");
00287         yield return www.SendWebRequest();
00288         if (www.result == UnityWebRequest.Result.ProtocolError)
00289         {
00290             Debug.Log("ERROR: cant get username. " + www.responseCode);
00291         }
00292         else
00293         {
00294             UserInfoStructResponse response =
00295                 JsonUtility.FromJson<UserInfoStructResponse>(www.downloadHandler.text);
00296             if (field != null) field.text = response.login;
00297 
00298         }
00299     }
00300 
00301     #endregion
00302 }
```

7.4 DataBase.cs

```
00001 using System.Collections.Generic;
00002 using System.Data;
00003 using UnityEngine;
00004 using Mono.Data.Sqlite;
00005 
00009 public class DataBase : MonoBehaviour
00010 {
00011     private static string dbName = "URI=file:VujiDB.db"; // Путь к файлу бд
00012     private static SqliteConnection _connection; // Вспомогательное поле для хранения сессии
00013     private static SqliteCommand _command; // Вспомогательное поле для хранения команды (запроса) SQL
00014     private static IDataReader _reader; // Вспомогательное поля для чтеня даннных
00015 
00016     #region Unity Methods
00017 
00018     void Start()
00019     {
00020         CreateDB();
00021     }
00022 
00023     #endregion
00024 
00025     #region Private Methods
00026 
00030     private static void OpenConnection()
00031     {
00032         _connection = new SqliteConnection(dbName);
00033         _command = new SqliteCommand(_connection);
00034         _connection.Open();
00035     }
00036 
00040     private static void CloseConnection()
00041     {
00042         _connection.Close();
00043         _command.Dispose();
00044     }
00045 
00049     private void CreateDB()
00050     {
00051         OpenConnection();
00052         _command.CommandText = @"PRAGMA foreign_keys=OFF;
00053                                  BEGIN TRANSACTION;
00054                                  CREATE TABLE token (id INTEGER, token STRING, PRIMARY KEY (id));
00055                                  CREATE TABLE keybinds (id INTEGER, name STRING, keybind STRING, category STRING,
      PRIMARY KEY (id));
00056                                  INSERT INTO keybinds VALUES(1,'EscapeMenu','Escape','UI');
00057                                  INSERT INTO keybinds VALUES(2,'Right','D','Movement');
00058                                  INSERT INTO keybinds VALUES(3,'Left','A','Movement');
00059                                  INSERT INTO keybinds VALUES(4,'Forward','W','Movement');
00060                                  INSERT INTO keybinds VALUES(5,'Backward','S','Movement');
00061                                  INSERT INTO keybinds VALUES(6,'Attack','Space','Ability');
00062                                  INSERT INTO keybinds VALUES(7,'Use Skill','Mouse0','Ability');
00063                                  INSERT INTO keybinds VALUES(8,'Skill 1','Q','Ability');
00064                                  INSERT INTO keybinds VALUES(9,'Skill 2','E','Ability');
00065                                  INSERT INTO keybinds VALUES(10,'Slot 1','Alpha1','Ability');
00066                                  INSERT INTO keybinds VALUES(11,'Slot 2','Alpha2','Ability');
00067                                  INSERT INTO keybinds VALUES(13,'Slot 3','Alpha3','Ability');
00068                                  INSERT INTO keybinds VALUES(14,'Slot 4','Alpha4','Ability');
00069                                  INSERT INTO keybinds VALUES(15,'Slot 5','Alpha5','Ability');
00070                                  INSERT INTO keybinds VALUES(16,'Slot 6','Alpha6','Ability');
00071                                  INSERT INTO keybinds VALUES(17,'Slot 7','Alpha7','Ability');
00072                                  INSERT INTO keybinds VALUES(18,'Slot 8','Alpha8','Ability');
00073                                  INSERT INTO keybinds VALUES(19,'Slot 9','Alpha9','Ability');
00074                                  CREATE TABLE settings (id INTEGER, name STRING, value STRING, PRIMARY KEY (id));
```

```
00075                           INSERT INTO settings VALUES(1,'FPS',359);
00076                           INSERT INTO settings VALUES(2,'VSync','True');
00077                           INSERT INTO settings VALUES(3,'Resolution',2);
00078                           INSERT INTO settings VALUES(4,'Fullscreen','False');
00079                           INSERT INTO settings VALUES(5,'General volume','1');
00080                           INSERT INTO settings VALUES(6,'Music volume','1');
00081                           INSERT INTO settings VALUES(7,'Environment volume','1');
00082                           COMMIT;
00083 ";
00084       _command.ExecuteNonQuery();
00085       CloseConnection();
00086   }
00087
00092   private bool TokenInDB()
00093   {
00094       OpenConnection();
00095       _command.CommandText = "SELECT * FROM token WHERE EXISTS (SELECT id FROM token WHERE id =
      1)";
00096       _reader = _command.ExecuteReader();
00097       if (_reader.Read())
00098       {
00099           CloseConnection();
00100           return true;
00101       }
00102       else
00103       {
00104           CloseConnection();
00105           return false;
00106       }
00107   }
00108
00109   #endregion
00110
00111   #region Public Methods
00112
00117   public string GetToken()
00118   {
00119       string token = "None";
00120       if (TokenInDB())
00121       {
00122           OpenConnection();
00123           _command.CommandText = "SELECT * FROM token WHERE id=1";
00124           _reader = _command.ExecuteReader();
00125           while (_reader.Read())
00126           {
00127               token = _reader["token"].ToString();
00128           }
00129
00130           CloseConnection();
00131       }
00132
00133       if (token == "")
00134       {
00135           token = "None";
00136       }
00137
00138       return token;
00139   }
00140
00145   public void SetToken(string token)
00146   {
00147       if (TokenInDB())
00148       {
00149           OpenConnection();
00150           _command.CommandText = "UPDATE token SET token = '" + token + "' WHERE id =1;";
00151           _command.ExecuteNonQuery();
00152           CloseConnection();
00153       }
00154       else
00155       {
00156           OpenConnection();
00157           _command.CommandText = "INSERT INTO token (token) VALUES ('" + token + "');";
00158           _command.ExecuteNonQuery();
00159           CloseConnection();
00160       }
00161   }
00167   public bool ExistKeybind(string name)
00168   {
00169       OpenConnection();
00170       _command.CommandText = "SELECT * from keybinds WHERE name ='" + name + "';";
00171       _reader = _command.ExecuteReader();
00172       if (_reader.Read())
00173       {
00174           CloseConnection();
00175           return true;
00176       }
00177       CloseConnection();
```

```
00178         return false;
00179
00180     }
00187     public void AddKeybind(string name, KeyCode key, string category)
00188     {
00189         OpenConnection();
00190         _command.CommandText = "INSERT INTO keybinds (name, keybind, category) VALUES ('" + name + "', '" +
      key.ToString() + "', '" + category + "');";
00191         _command.ExecuteNonQuery();
00192         CloseConnection();
00193     }
00199     public void SetKeybind(string name, KeyCode key)
00200     {
00201         if (!ExistKeybind(name)) { return; }
00202         OpenConnection();
00203         _command.CommandText = "UPDATE keybinds SET keybind  = '" + key.ToString() + "' WHERE name ='" +
      name + "';";
00204         _command.ExecuteNonQuery();
00205         CloseConnection();
00206     }
00211     public List<Keybind> GetKeybinds()
00212     {
00213         List<Keybind> keybinds = new List<Keybind>();
00214         OpenConnection();
00215         _command.CommandText = "SELECT * from keybinds";
00216         _reader = _command.ExecuteReader();
00217         while (_reader.Read())
00218         {
00219             keybinds.Add(new Keybind(_reader["name"].ToString(), _reader["keybind"].ToString(),
      _reader["category"].ToString()));
00220         }
00221         CloseConnection();
00222         return keybinds;
00223     }
00229     public void AddSetting(string name, string value)
00230     {
00231         OpenConnection();
00232         _command.CommandText = "INSERT INTO settings (name, value) VALUES ('" + name + "', '" + value + "');";
00233         _command.ExecuteNonQuery();
00234         CloseConnection();
00235     }
00241     public bool ExistSetting(string name)
00242     {
00243         OpenConnection();
00244         _command.CommandText = "SELECT * from settings WHERE name ='" + name + "';";
00245         _reader = _command.ExecuteReader();
00246         if (_reader.Read())
00247         {
00248             CloseConnection();
00249             return true;
00250         }
00251         CloseConnection();
00252         return false;
00253     }
00259     public void SetSetting(string name, string value)
00260     {
00261         if (!ExistSetting(name)) { return; }
00262         OpenConnection();
00263         _command.CommandText = "UPDATE settings SET value  = '" + value + "' WHERE name ='" + name + "';";
00264         _command.ExecuteNonQuery();
00265         CloseConnection();
00266     }
00271     public List<Setting> GetSettings()
00272     {
00273         List<Setting> keybinds = new List<Setting>();
00274         OpenConnection();
00275         _command.CommandText = "SELECT * from settings";
00276         _reader = _command.ExecuteReader();
00277         while (_reader.Read())
00278         {
00279             keybinds.Add(new Setting(_reader["name"].ToString(), _reader["value"].ToString()));
00280         }
00281         CloseConnection();
00282         return keybinds;
00283     }
00284
00285     #endregion
00286 }
00287
00291 public class Keybind
00292 {
00293     public string name; // Название действия
00294     public string key; // Ключ действия
00295     public string category; // Категория действия
00302     public Keybind(string name, string key, string category)
00303     {
00304         this.name = name;
```

```
00305         this.key = key;
00306         this.category = category;
00307     }
00308 }
00312 public class Setting
00313 {
00314     public string name; // Название настройки
00315     public string value; // Значение
00321     public Setting(string name, string value)
00322     {
00323         this.name = name;
00324         this.value = value;
00325     }
00326 }
```

7.5 AgressionTrigger.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 public class AgressionTrigger : MonoBehaviour
00006 {
00007     void OnTriggerEnter2D(Collider2D other)
00008     {
00009         if (other.CompareTag("Player"))
00010         {
00011             transform.parent.GetComponent<EnemyAI>().AgressionStart(other.gameObject);
00012         }
00013     }
00014 }
```

7.6 AttackTrigger.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 public class AttackTrigger : MonoBehaviour
00006 {
00007     void OnTriggerStay2D(Collider2D other)
00008     {
00009         if (other.CompareTag("Player"))
00010         {
00011             transform.parent.GetComponent<EntityMelee>().Attack(other.gameObject);
00012         }
00013     }
00014 }
```

7.7 EnemyAI.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 using Pathfinding;
00005
00006 public class EnemyAI : MonoBehaviour
00007 {
00008     GameObject target;
00009     [SerializeField] float nextWaypointDistance = 3f;
00010
00011     Path path;
00012     int currentWaypoint = 0;
00013     public bool reachedEndOfPath = false;
00014
00015     Seeker seeker;
00016     Rigidbody2D rb;
00017
00018     void Start()
00019     {
00020         seeker = GetComponent<Seeker>();
00021         rb = GetComponent<Rigidbody2D>();
00022     }
00023
00024     public void AgressionStart(GameObject target)
00025     {
```

```
00026         this.target = target;
00027         InvokeRepeating("UpdatePath", 0f, 0.5f);
00028     }
00029 
00030     void UpdatePath()
00031     {
00032 
00033 
00034         if (seeker.IsDone())
00035             seeker.StartPath(rb.position, target.transform.position, OnPathComplete);
00036     }
00037 
00038     void OnPathComplete(Path p)
00039     {
00040         if(!p.error)
00041         {
00042             path = p;
00043             currentWaypoint = 0;
00044         }
00045     }
00046 
00047     void FixedUpdate()
00048     {
00049         if(target == null)
00050             return;
00051 
00052         if(path == null)
00053             return;
00054 
00055         if(currentWaypoint >= path.vectorPath.Count)
00056         {
00057             reachedEndOfPath = true;
00058             return;
00059         }
00060         else
00061         {
00062             reachedEndOfPath = false;
00063         }
00064 
00065         Vector2 direction = ((Vector2)path.vectorPath[currentWaypoint] - rb.position).normalized;
00066         Vector2 force = direction * GetComponent<BaseEntity>().GetMoveSpeed() * Time.deltaTime;
00067 
00068         rb.AddForce(force);
00069 
00070         float distance = Vector2.Distance(rb.position, path.vectorPath[currentWaypoint]);
00071 
00072         if(distance < nextWaypointDistance)
00073         {
00074             currentWaypoint++;
00075         }
00076     }
00077 
00078 }
```

7.8 EntityMelee.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 
00005 public class EntityMelee : MonoBehaviour
00006 {
00007     [SerializeField] private LayerMask enemyLayers;
00008     [SerializeField] private float attackDistance = 1f;
00009     [SerializeField] private float attackRange = 1f;
00010     [SerializeField] float attackTimeout;
00011 
00012     private bool _isTimeout = false;
00013 
00014     private int _damage;
00015     private Vector3 _attackPoint;
00016 
00017     private void Start() {
00018         _damage = gameObject.GetComponent<BaseEntity>().GetBaseDamage();
00019     }
00020 
00021     public void Attack(GameObject target)
00022     {
00023         if (!_isTimeout)
00024         {
00025             StartCoroutine("AttackTiemout");
00026 
00027             var xLen = target.transform.position.x - transform.position.x;
00028             var yLen = target.transform.position.y - transform.position.y;
```

```
00029           var xyLen = (float) (Mathf.Sqrt(Mathf.Pow(xLen, 2) + Mathf.Pow(yLen, 2)));
00030           var x = (xLen * attackDistance) / xyLen + transform.position.x;
00031           var y = (yLen * attackDistance) / xyLen + transform.position.y;
00032
00033           _attackPoint = new Vector3(x, y, 0);
00034
00035           Collider2D[] hitEnemies = Physics2D.OverlapCircleAll(_attackPoint, attackRange, enemyLayers);
00036
00037           foreach (Collider2D enemy in hitEnemies)
00038           {
00039               GameObject enemyGameObject = enemy.transform.parent.gameObject;
00040               if (enemyGameObject != gameObject)
00041                   enemyGameObject.GetComponent<BaseEntity>().TakeDamage(_damage);
00042           }
00043       }
00044   }
00045
00046   IEnumerator AttackTiemout()
00047   {
00048       _isTimeout = true;
00049       yield return new WaitForSeconds(attackTimeout);
00050       _isTimeout = false;
00051   }
00052 }
```

7.9 PlayerAOE.cs

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using UnityEngine;
00005 using Photon.Pun;
00006
00007 public class PlayerAOE : MonoBehaviour
00008 {
00009
00010     [SerializeField] Transform firePoint;
00011
00012     [Serializable]
00013     public struct AOEstruct
00014     {
00015         public string AOEKey;
00016         public GameObject AOE;
00017     }
00018     [SerializeField] private AOEstruct[] AOEs;
00019     private Dictionary<string, GameObject> _allAOEs = new Dictionary<string, GameObject>();
00020
00021     private PhotonView _view;
00022
00023     void Start()
00024     {
00025         _view = gameObject.GetComponent<PhotonView>();
00026
00027         // Заполнение нормального словаря всех проджектайлов
00028         if (AOEs.Length != 0)
00029             for (int i = 0; i < AOEs.Length; i++)
00030             {
00031                 Debug.Log(AOEs[i].AOEKey + " " + AOEs[i].AOE);
00032                 this._allAOEs[AOEs[i].AOEKey] = AOEs[i].AOE;
00033             }
00034     }
00035
00036     public void Attack(string AOEKey)
00037     {
00038         _view.RPC("CreateAOE", RpcTarget.All, AOEKey);
00039     }
00040
00041     [PunRPC]
00042     private void CreateAOE(string AOEKey)
00043     {
00044         GameObject AOE = _allAOEs[AOEKey];
00045         BaseAOE aoeBase = AOE.GetComponent<BaseAOE>();
00046         aoeBase.SetSenderCollider(this.gameObject);
00047
00048         Instantiate(AOE, firePoint.position, Quaternion.identity);
00049     }
00050 }
```

7.10 PlayerMelee.cs

```
00001 using System;
```

```
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using UnityEngine;
00005 using Photon.Pun;
00006 using Photon.Pun.UtilityScripts;
00007
00008 public class PlayerMelee : MonoBehaviour
00009 {
00010     [SerializeField] private LayerMask enemyLayers;
00011     [SerializeField] private int damage = 10;
00012     [SerializeField] private float attackDistance = 1f;
00013     [SerializeField] private float attackRange = 1f;
00014     [SerializeField] private float attackTimeout = 1f;
00015
00016     private Vector3 _attackPoint;
00017     private Vector3 _playerPosition;
00018     private Vector3 _mousePosition;
00019     private PhotonView _myView;
00020     private bool _isTimeout = false;
00021
00022     private Animator _animator;
00023
00024
00025
00026     private void Start()
00027     {
00028         _myView = gameObject.GetComponent<PhotonView>();
00029     }
00030
00031     private void Update()
00032     {
00033         if (_myView.IsMine)
00034         {
00035             _mousePosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);
00036             _playerPosition = transform.position;
00037             _mousePosition.z = 0;
00038             var xLen = _mousePosition.x - _playerPosition.x;
00039             var yLen = _mousePosition.y - _playerPosition.y;
00040             var xyLen = (float)(Math.Sqrt(Math.Pow(xLen, 2) + Math.Pow(yLen, 2)));
00041             var x = (xLen * attackDistance) / xyLen + _playerPosition.x;
00042             var y = (yLen * attackDistance) / xyLen + _playerPosition.y;
00043             _attackPoint = new Vector3(x, y, 0);
00044         }
00045     }
00046     [PunRPC]
00047     public void MasterCheckMeleeAttack()
00048     {
00049         Debug.Log("Mellee check");
00050         if(_isTimeout) return;
00051         StartCoroutine(AttackTiemout());
00052
00053         Collider2D[] hitEnemies = Physics2D.OverlapCircleAll(_attackPoint, attackRange, enemyLayers);
00054
00055         foreach (Collider2D enemy in hitEnemies)
00056         {
00057             if (CanDamageThisEnemy(enemy))
00058             {
00059                 int dmg = damage + gameObject.GetComponent<BaseEntity>().GetBaseDamage();
00060                 _myView.RPC("TakeDamageRemote", RpcTarget.All,
    enemy.GetComponentInParent<PhotonView>().ViewID, dmg);
00061             }
00062         }
00063     }
00064
00065
00066     private bool CanDamageThisEnemy(Collider2D enemyCollider)
00067     {
00068         GameObject enemyGameObject = enemyCollider.transform.parent.gameObject;
00069
00070         if (enemyGameObject == gameObject)
00071         {
00072             return false;
00073         }
00074
00075         if (enemyGameObject.CompareTag("Player"))
00076         {
00077             var otherPlayerView = enemyGameObject.GetComponent<PhotonView>();
00078
00079             if (otherPlayerView.Owner.GetPhotonTeam().Name == _myView.Owner.GetPhotonTeam().Name)
00080             {
00081                 return false;
00082             }
00083         }
00084
00085         return true;
00086     }
00087
```

```
00088     private IEnumerator AttackTiemout()
00089     {
00090         _isTimeout = true;
00091         yield return new WaitForSeconds(attackTimeout);
00092         _isTimeout = false;
00093     }
00094 }
```

7.11 PlayerProjectile.cs

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using UnityEngine;
00005 using Photon.Pun;
00006
00007 public class PlayerProjectile : MonoBehaviour
00008 {
00009     #region Fields
00010
00011     [SerializeField] private GameObject projectile;
00012     [SerializeField] private Transform rotatePoint;
00013     [SerializeField] private Transform firePoint;
00014
00015
00016     [Serializable]
00017     public struct Projectile
00018     {
00019         public string projectileKey;
00020         public GameObject projectile;
00021     }
00022     [SerializeField] private Projectile[] projectiles;
00023     private Dictionary<string, GameObject> _allProjectiles = new Dictionary<string, GameObject>();
00024
00025     private Vector3 _mousePosition;
00026     private Vector3 _aimDirection;
00027     private float _aimAngle;
00028     private PhotonView _view;
00029     private bool isTimeout = false;
00030     #endregion
00031
00032     private void Start()
00033     {
00034         _view = GetComponent<PhotonView>();
00035
00036         // Заполнение нормального словаря всех проджектайлов
00037         if (projectiles.Length != 0)
00038             for (int i = 0; i < projectiles.Length; i++)
00039             {
00040                 Debug.Log(projectiles[i].projectileKey + " " + projectiles[i].projectile);
00041                 this._allProjectiles[projectiles[i].projectileKey] = projectiles[i].projectile;
00042             }
00043     }
00044
00045     public void Attack(string projectileKey)
00046     {
00047         //RemoteProjectileAttack(_aimAngle, _aimDirection, projectileKey);
00048         _view.RPC("RemoteProjectileAttack", RpcTarget.All, _aimAngle, _aimDirection, projectileKey);
00049     }
00050
00051     public void Attack(string projectileKey, float time)
00052     {
00053         //RemoteProjectileAttack(_aimAngle, _aimDirection, projectileKey);
00054         if(!isTimeout){
00055             StartCoroutine(AttackTimeout(time));
00056             _view.RPC("RemoteProjectileAttack", RpcTarget.All, _aimAngle, _aimDirection, projectileKey);
00057         }
00058     }
00059
00060     private void Update()
00061     {
00062         if (_view.IsMine)
00063         {
00064             _mousePosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);
00065             _aimDirection = _mousePosition - rotatePoint.position;
00066             _aimAngle = Mathf.Atan2(_aimDirection.y, _aimDirection.x) * Mathf.Rad2Deg;
00067             rotatePoint.rotation = Quaternion.Euler(0, 0, _aimAngle);
00068         }
00069     }
00070
00071     [PunRPC]
00072     private void RemoteProjectileAttack(float aimAngle, Vector3 aimDirection, string projectileKey)
00073     {
00074         projectile = _allProjectiles[projectileKey];
```

```
00075         BaseProjectile projectileBase = projectile.GetComponent<BaseProjectile>();
00076
00077         projectileBase.SetAimDirection(aimDirection);
00078         projectileBase.SetAimAngel(aimAngle);
00079         projectileBase.SetSenderCollider(gameObject);
00080         projectileBase.AddDamage(gameObject.GetComponent<BaseEntity>().GetBaseDamage());
00081
00082         GameObject projectileInst = Instantiate(projectile, firePoint.position, firePoint.rotation);
00083     }
00084
00085
00086     private IEnumerator AttackTimeout(float time)
00087     {
00088         isTimeout = true;
00089         yield return new WaitForSeconds(time);
00090         isTimeout = false;
00091     }
00092 }
```

7.12 BaseAOE.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using Photon.Pun;
00004 using Photon.Pun.UtilityScripts;
00005 using UnityEngine;
00006
00007 public class BaseAOE : MonoBehaviour
00008 {
00009     #region Fields
00010
00011     [SerializeField] private GameObject effect;
00012     [SerializeField] private string aoeName = "New Projectile";
00013     [SerializeField] private string aoeDescription = "";
00014     [SerializeField] private float lifeTime = 5.0f;
00015     [SerializeField] private GameObject senderGameObject;
00016
00017     // список тегов которым должен наноситься урон и где проджектайл должен уничтожаться
00018     private List<string> _damageTags = new List<string>() {"Entity", "Player"};
00019     private List<string> _destroyTags = new List<string>() {"Entity", "Player", "Wall"};
00020
00021     #endregion
00022
00023     #region Public Methods
00024
00025     public void SetSenderCollider(GameObject senderObject)
00026     {
00027         senderGameObject = senderObject;
00028     }
00029
00030     public string GetAoeName()
00031     {
00032         return aoeName;
00033     }
00034
00035     public string GetAoeDescription()
00036     {
00037         return aoeDescription;
00038     }
00039
00040     public void SetDestroyTags(List<string> newDestroyTags)
00041     {
00042         _destroyTags = newDestroyTags;
00043     }
00044
00045     public void SetDamageTags(List<string> newDamageTags)
00046     {
00047         _damageTags = newDamageTags;
00048     }
00049
00050
00051     #endregion
00052
00053     void Start()
00054     {
00055         StartCoroutine(DestroyAfterLifeTime());
00056     }
00057
00058     IEnumerator DestroyAfterLifeTime()
00059     {
00060         yield return new WaitForSeconds(lifeTime);
00061         Destroy(gameObject);
00062     }
00063
```

```
00064     void OnTriggerEnter2D(Collider2D colliderObject)
00065     {
00066         GameObject enemyGameObject = colliderObject.gameObject;
00067         if (CanDamageThisEntity(enemyGameObject) && PhotonNetwork.IsMasterClient)
00068         {
00069             colliderObject.gameObject.GetComponent<BaseEntity>().AddEffect(effect);
00070         }
00071     }
00072 
00081     private bool CanDamageThisEntity(GameObject enemyGameObject)
00082     {
00083         if (!_damageTags.Contains(enemyGameObject.tag))
00084         {
00085             return false;
00086         }
00087 
00088         if (enemyGameObject == senderGameObject)
00089         {
00090             return false;
00091         }
00092 
00093         if (enemyGameObject.CompareTag("Player"))
00094         {
00095             var otherPlayerView = enemyGameObject.GetComponent<PhotonView>();
00096 
00097             if (otherPlayerView.Owner.GetPhotonTeam().Name ==
00098                 senderGameObject.GetComponent<PhotonView>().Owner.GetPhotonTeam().Name)
00099             {
00100                 return false;
00101             }
00102         }
00103 
00104         return true;
00105     }
00106 }
```

7.13 BaseEffect.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 
00005 public abstract class BaseEffect : MonoBehaviour
00006 {
00007     [SerializeField] public string effectName = "New Effect";
00008     [SerializeField] public string description = "Effect description";
00009     [SerializeField] public float duration = 5f;
00010 
00011     [SerializeField] public Sprite effectSprite;
00012 
00013     public virtual void ApplyEffect(GameObject entity)
00014     {
00015         Debug.Log("Apply Effect " + effectName + " on entity " + entity.name);
00016     }
00017 
00018 }
```

7.14 BaseEntity.cs

```
00001 using System.Collections.Generic;
00002 using System;
00003 using Photon.Pun;
00004 using Photon.Pun.UtilityScripts;
00005 using UnityEngine;
00006 using UnityEngine.UI;
00007 using Random = UnityEngine.Random;
00008 
00009 public class BaseEntity : MonoBehaviour
00010 {
00011     #region Entity Stats
00012     [SerializeField] private string entityName = "baseEntityName";
00013     [SerializeField] private int baseDamage = 5;
00014     [SerializeField] private int defense = 0;
00015     [SerializeField] private float healthPoints = 100.0f;
00016     [SerializeField] private float maxHealthPoints = 100.0f;
00017     [SerializeField] private float moveSpeed = 3.0f;
00018     [SerializeField] private float energy = 100.0f;
00019     [SerializeField] private float maxEnergy = 100.0f;
00020     [SerializeField] private float healthRegeneration = 1.0f;
00021     [SerializeField] private float energyRegeneration = 5.0f;
```

```
00022     #endregion
00023
00024     #region Skills
00025     //Аналог словаря для юнити инспектора
00026     [Serializable]
00027     public struct Skill
00028     {
00029         public string key;
00030         public GameObject skill;
00031     }
00032     public Skill[] skills;
00033     private Dictionary<string, GameObject> _skills = new Dictionary<string, GameObject>();
00034
00035     public Action<string, bool> OnSkillSelectionChange;
00036
00037     #endregion
00038
00039     #region Private fields
00040     // Обязятальный префаб для выпадения предметов
00041     public GameObject _droppedItemPrefab;
00042
00043     private PhotonView _view;
00044
00045     private float _regenerationTick = 1;
00046     private float _currentTick;
00047     private bool _isSkill1Cooldown = false;
00048     private bool _isSkill2Cooldown = false;
00049     private string _selectedSkill = "";
00050     #endregion
00051
00052     #region DisplayedInformation
00053     [SerializeField] public HealthBarManager healthBar;
00054     [SerializeField] public EntityNameManager displayedName;
00055     public static Action<BaseEntity, string> teamSpawn;
00056     public Controllers _controller;
00057
00058     public bool isDead { get; private set; } = false;
00059
00060     public Action<BaseEffect, BaseEntity> OnEffectApply;
00061     #endregion
00062
00063     private void Start()
00064     {
00065         _view = gameObject.GetComponent<PhotonView>();
00066         _currentTick = _regenerationTick;
00067
00068         // Заполнение обычного словаря скилов из словаря из инспектора
00069         if (skills.Length != 0)
00070             for (int i = 0; i < skills.Length; i++)
00071             {
00072                 Debug.Log(skills[i].key + " " + skills[i].skill);
00073                 this._skills[skills[i].key] = skills[i].skill;
00074             }
00075
00076
00077         maxHealthPoints = Mathf.Max(maxHealthPoints, healthPoints);
00078         maxEnergy = Mathf.Max(maxEnergy, energy);
00079         float height = 1.0f;
00080         healthBar.SetOffset(new Vector3(0, height * 0.6f, 0));
00081         healthBar.SetHealth(healthPoints, maxHealthPoints);
00082         displayedName.SetOffset(new Vector3(0, height * 0.6f, 0));
00083         if (gameObject.CompareTag("Player"))
00084         {
00085             if (_view.IsMine)
00086             {
00087                 _view.RPC("UpdateText", RpcTarget.All, "[" + PhotonNetwork.LocalPlayer.GetPhotonTeam().Name + "] ",
      PhotonNetwork.LocalPlayer.NickName);
00088             }
00089         }
00090
00091     }
00092
00093     private void Update()
00094     {
00095         healthBar.SetHealth(healthPoints, maxHealthPoints);
00096     }
00097
00098     [PunRPC]
00099     public void UpdateText(string teamTag, string newText)
00100     {
00101         GetComponentInChildren<Text>().text =  teamTag + newText;
00102         if ("[" + PhotonNetwork.LocalPlayer.GetPhotonTeam().Name + "] " == teamTag) teamSpawn?.Invoke(this,
      newText);
00103     }
00104
00105     [PunRPC]
00106     public void IncreaseMaxHealth(float toAdd)
```

```
00107     {
00108         maxHealthPoints += toAdd;
00109     }
00110     [PunRPC]
00111     public void IncreaseMaxEnergy(float toAdd)
00112     {
00113         maxEnergy += toAdd;
00114     }
00115     [PunRPC]
00116     public void Heal(float hp) {
00117         healthPoints = healthPoints + hp > maxHealthPoints ? maxHealthPoints : healthPoints + hp;
00118     }
00119 
00120     [PunRPC]
00121     void IncreasePoints(){
00122         healthPoints = healthPoints + healthRegeneration > maxHealthPoints?maxHealthPoints:healthPoints +
    healthRegeneration;
00123         energy = energy + energyRegeneration > maxEnergy?maxEnergy:energy + energyRegeneration;
00124     }
00125 
00126     #region Public Methods
00127 
00128     public void AddEffect(GameObject effect)
00129     {
00130         OnEffectApply?.Invoke(effect.GetComponent<BaseEffect>(), this);
00131         effect.GetComponent<BaseEffect>().ApplyEffect(this.gameObject);
00132     }
00133 
00134     public void TickPoints()
00135     {
00136         _currentTick -= Time.deltaTime;
00137         if (_currentTick <= 0)
00138         {
00139             _view.RPC("IncreasePoints", RpcTarget.All);
00140             _currentTick = _regenerationTick;
00141         }
00142     }
00143 
00144     public void UseSkill()
00145     {
00146         if (_selectedSkill.StartsWith("Skill"))
00147         {
00148             Debug.Log("Used " + _selectedSkill);
00149             StartCoroutine(_skills[_selectedSkill].GetComponent<BaseSkill>().UseSkill(this.gameObject, _selectedSkill));
00150             deSelectSkill();
00151         }
00152         else
00153         {
00154             Debug.Log("Skill is not selected");
00155         }
00156     }
00157 
00158     public void selectSkill(string skillName)
00159     {
00160         if ((skillName == "Skill 1" && !_isSkill1Cooldown) || (skillName == "Skill 2" && !_isSkill2Cooldown))
00161         {
00162             Debug.Log("Selected " + skillName);
00163             this._selectedSkill = skillName;
00164             OnSkillSelectionChange?.Invoke(_selectedSkill, true);
00165         }
00166         else
00167         {
00168             Debug.Log(skillName + " COOLDOWN");
00169         }
00170     }
00171 
00172     public void deSelectSkill()
00173     {
00174         Debug.Log("Deselected" + _selectedSkill);
00175         OnSkillSelectionChange?.Invoke(_selectedSkill, false);
00176         this._selectedSkill = "";
00177     }
00178 
00179     public string GetSelectedSkill()
00180     {
00181         return _selectedSkill;
00182     }
00183 
00184     public void setIsCooldown(string key, bool value)
00185     {
00186         if (key == "Skill 1") this._isSkill1Cooldown = value;
00187         if (key == "Skill 2") this._isSkill2Cooldown = value;
00188     }
00189 
00190 
00191     public float GetMoveSpeed()
00192     {
```

```
00193         return moveSpeed;
00194     }
00195 
00196     public float GetHealthPoints()
00197     {
00198         return healthPoints;
00199     }
00200 
00201     public float GetMaxHealthPoints()
00202     {
00203         return maxHealthPoints;
00204     }
00205 
00206     public float GetEnergyPoints()
00207     {
00208         return energy;
00209     }
00210 
00211     public float GetMaxEnergyPoints()
00212     {
00213         return maxEnergy;
00214     }
00215 
00216     public int GetBaseDamage()
00217     {
00218         return baseDamage;
00219     }
00220 
00221     public string GetEntityName()
00222     {
00223         return entityName;
00224     }
00225 
00226     public int GetDefense()
00227     {
00228         return defense;
00229     }
00230 
00231 
00232     public void IncreaseDamage(int addDamage)
00233     {
00234         this.baseDamage += addDamage;
00235     }
00236 
00237     public void DecreaseDamage(int addDamage)
00238     {
00239         this.baseDamage -= addDamage;
00240     }
00241 
00242     public void IncreaseSpeed(int addSpeed)
00243     {
00244         this.moveSpeed += addSpeed;
00245     }
00246 
00247     public void DecreaseSpeed(int addSpeed)
00248     {
00249         this.moveSpeed -= addSpeed;
00250     }
00251 
00252     public void IncreaseDefense(int addDefense)
00253     {
00254         this.defense += addDefense;
00255     }
00256 
00257     public void DecreaseDefense(int addDefense)
00258     {
00259         this.defense -= addDefense;
00260     }
00261 
00262     public bool spendEnergy(float energyCost)
00263     {
00264         if (energyCost > this.energy) return false;
00265 
00266         this.energy -= energyCost;
00267         return true;
00268     }
00269 
00270     public void TakeDamage(int healthDamage)
00271     {
00272         // Проверка на полное поглощение урона
00273         if(defense - healthDamage >= 0) return;
00274 
00275         healthPoints = healthPoints - healthDamage + defense;
00276         if (healthPoints <= 0)
00277         {
00278             Death();
00279         }
```

```
00280
00281         Debug.Log(entityName + " hp is " + healthPoints);
00282     }
00283     #endregion
00284     private void Death()
00285     {
00286         isDead = true;
00287         DropAllItems();
00288         if(gameObject.CompareTag("Player"))
00289             gameObject.GetComponent<PlayerScript>().KillPlayer();
00290         else
00291             PhotonNetwork.Destroy(gameObject);
00292     }
00293
00294     private void DropAllItems()
00295     {
00296         Inventory inventory = GetComponent<Inventory>();
00297         var items = inventory.GetAllItems();
00298         foreach (BaseItem itemData in items)
00299         {
00300             Vector2 position = new Vector2(gameObject.transform.position.x,gameObject.transform.position.y - 0.2f);
00301
00302
00303             GameObject droppedItem = Instantiate(_droppedItemPrefab, position, Quaternion.identity);
00304             droppedItem.GetComponent<DroppedItem>().SetItem(itemData);
00305         }
00306     }
00307
00308     [PunRPC]
00309     private void TakeDamageRemote(int photonID, int newDamage)
00310     {
00311         GameObject obj = PhotonView.Find(photonID).gameObject;
00312         obj.GetComponent<BaseEntity>().TakeDamage(newDamage);
00313     }
00314
00315 }
```

7.15 BaseItem.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 [CreateAssetMenu(fileName = "Item", menuName = "ScriptableObject/Item")]
00006 public class BaseItem : ScriptableObject
00007 {
00008     [SerializeField] private string itemName = "Item";
00009     [SerializeField] private string description = "Basic Item";
00010     [SerializeField] private int amount = 1;
00011     [SerializeField] private Sprite image;
00012
00013     public void SetAmount(int newAmount)
00014     {
00015         if (newAmount < 0)
00016         {
00017             return;
00018         }
00019
00020         amount = newAmount;
00021     }
00022
00023     public string GetItemName()
00024     {
00025         return itemName;
00026     }
00027
00028     public string GetDescription()
00029     {
00030         return description;
00031     }
00032
00033     public int GetAmount()
00034     {
00035         return amount;
00036     }
00037
00038     public Sprite GetImage()
00039     {
00040         return image;
00041     }
00042     public virtual bool UseItem(GameObject owner)
00043     {
00044         // Do smth
00045         amount = amount - 1;
```

```
00046         if (amount < 1)
00047         {
00048             return false;
00049         }
00050         return true;
00051     }
00052 }
```

7.16 BasePassiveSkill.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 public class BasePassiveSkill : MonoBehaviour
00006 {
00007
00008     [Tooltip("Название способности")]
00009     [SerializeField] protected string skillName;
00010
00011     [Tooltip("Описание способности")]
00012     [SerializeField] protected string skillDescription;
00013
00014     public string GetName()
00015     {
00016         return skillName;
00017     }
00018
00019     public string GetDescription()
00020     {
00021         return skillDescription;
00022     }
00023
00024     public virtual void ActivatePassiveSkill(GameObject caster) {
00025         Debug.Log(caster.GetComponent<BaseEntity>().GetEntityName() + " activated passive skill");
00026     }
00027 }
```

7.17 BaseProjectile.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using Photon.Pun;
00004 using Photon.Pun.UtilityScripts;
00005 using UnityEngine;
00006
00007 public class BaseProjectile : MonoBehaviour
00008 {
00009     #region Fields
00010
00011     [SerializeField] private string projectileName = "New Projectile";
00012     [SerializeField] private string projectileDescription = "";
00013     [SerializeField] private int projectileDamage = 1;
00014     [SerializeField] private float velocity = 1.0f;
00015     [SerializeField] private float lifeTime = 5.0f;
00016     [SerializeField] private bool destroyOnCollide = true;
00017     [SerializeField] private Vector3 aimDirection;
00018     [SerializeField] private float aimAngle;
00019     [SerializeField] private GameObject senderGameObject;
00020
00021     // список тегов которым должен наноситься урон и где проджектайл должен уничтожаться
00022     private List<string> _damageTags = new List<string>() {"Entity", "Player"};
00023     private List<string> _destroyTags = new List<string>() {"Entity", "Player", "Wall"};
00024     private Rigidbody2D _projectileRb2d;
00025
00026     #endregion
00027
00028     #region Public Methods
00029
00030     public void SetSenderCollider(GameObject senderObject)
00031     {
00032         senderGameObject = senderObject;
00033     }
00034
00035     public string GetProjectileName()
00036     {
00037         return projectileName;
00038     }
00039
00040     public string GetProjectileDescription()
```

```
00041     {
00042         return projectileDescription;
00043     }
00044 
00045     public void SetAimDirection(Vector3 newAimDirection)
00046     {
00047         aimDirection = newAimDirection;
00048     }
00049 
00050     public void SetAimAngel(float newAimAngel)
00051     {
00052         aimAngle = newAimAngel;
00053     }
00054 
00055     public void SetDestroyTags(List<string> newDestroyTags)
00056     {
00057         _destroyTags = newDestroyTags;
00058     }
00059 
00060 
00061     public void SetDamageTags(List<string> newDamageTags)
00062     {
00063         _damageTags = newDamageTags;
00064     }
00065 
00066     public void AddDamage(int additionalDamage)
00067     {
00068         this.projectileDamage += additionalDamage;
00069     }
00070 
00071     #endregion
00072 
00073     void Start()
00074     {
00075         _projectileRb2d = GetComponent<Rigidbody2D>();
00076         SendProjectile();
00077         StartCoroutine(DestroyAfterLifeTime());
00078     }
00079 
00080     void SendProjectile()
00081     {
00082         _projectileRb2d.velocity = new Vector2(aimDirection.x, aimDirection.y).normalized * velocity;
00083         transform.rotation = Quaternion.Euler(0, 0, aimAngle);
00084     }
00085 
00086     IEnumerator DestroyAfterLifeTime()
00087     {
00088         yield return new WaitForSeconds(lifeTime);
00089         Destroy(gameObject);
00090     }
00091 
00092     void OnTriggerEnter2D(Collider2D colliderObject)
00093     {
00094         GameObject enemyGameObject = colliderObject.gameObject;
00095         if (colliderObject.gameObject.layer == 9)
00096         {
00097             enemyGameObject = colliderObject.transform.parent.gameObject;
00098             if (CanDamageThisEntity(enemyGameObject) && PhotonNetwork.IsMasterClient)
00099             {
00100                 senderGameObject.GetComponent<PhotonView>()
00101                     .RPC("TakeDamageRemote", RpcTarget.All, enemyGameObject.GetComponent<PhotonView>().ViewID,
00102                         projectileDamage);
00103             }
00104         }
00105         // Contains как мне кажется не лучший метод для "содержиться ли элемент в массиеве"
00106         if (destroyOnCollide & _destroyTags.Contains(enemyGameObject.tag) &
00107             (senderGameObject != enemyGameObject))
00108         {
00109             Destroy(gameObject);
00110         }
00111     }
00112 
00121     private bool CanDamageThisEntity(GameObject enemyGameObject)
00122     {
00123         if (!_damageTags.Contains(enemyGameObject.tag))
00124         {
00125             return false;
00126         }
00127 
00128         if (enemyGameObject == senderGameObject)
00129         {
00130             return false;
00131         }
00132 
00133         if (enemyGameObject.CompareTag("Player"))
00134         {
00135             var otherPlayerView = enemyGameObject.GetComponent<PhotonView>();
```

```
00136
00137             if (otherPlayerView.Owner.GetPhotonTeam().Name ==
00138                 senderGameObject.GetComponent<PhotonView>().Owner.GetPhotonTeam().Name)
00139             {
00140                 return false;
00141             }
00142         }
00143
00144         return true;
00145     }
00146 }
```

7.18 BaseSkill.cs

```
00001 using System;
00002 using System.Collections;
00003 using UnityEngine;
00004
00005
00006 [System.Serializable]
00007 public class BaseSkill : MonoBehaviour
00008 {
00009     [Tooltip("Название способности")]
00010     [SerializeField] protected string skillName;
00011
00012     [Tooltip("Описание способности")]
00013     [SerializeField] protected string skillDescription;
00014
00015     [Tooltip("Количество маны требуемое для использования способности")]
00016     [SerializeField] protected int energyCost;
00017
00018     [Tooltip("Время, которое проходит между нажатием кнопки и активацией способности")]
00019     [SerializeField] protected float castTime;
00020
00021     [Tooltip("Задержка перед повторным использованием способности")]
00022     [SerializeField] protected float cooldown;
00023
00024     [Tooltip("Запрет на движение игрока во время использования способности")]
00025     [SerializeField] protected bool cancelMovementOnCast;
00026
00027     [Tooltip("Skill Sprite")]
00028     [SerializeField] protected Sprite skillSprite;
00029
00030     public string GetName()
00031     {
00032         return skillName;
00033     }
00034
00035     public string GetDescription()
00036     {
00037         return skillDescription;
00038     }
00039     public int GetCost()
00040     {
00041         return energyCost;
00042     }
00043
00044     public float GetCastTime()
00045     {
00046         return castTime;
00047     }
00048
00049     public float GetCooldownTime()
00050     {
00051         return cooldown;
00052     }
00053
00054     public Sprite GetSprite()
00055     {
00056         return skillSprite;
00057     }
00058
00059     public static Action<float> onCast;
00060     public Action<float> onRelease;
00061
00062     public virtual IEnumerator UseSkill(GameObject caster, string key) {
00063         yield return new WaitForSeconds(0.0f);
00064     }
00065 }
```

7.19 BaseUpgrade.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 public class BaseUpgrade : MonoBehaviour
00006 {
00007     [SerializeField] private string upgradeName;
00008     [SerializeField] private string upgradeDescription;
00009     [SerializeField] private int upgradeCost;
00010
00011     public string GetName()
00012     {
00013         return upgradeName;
00014     }
00015     public string GetDescription()
00016     {
00017         return upgradeDescription;
00018     }
00019     public int GetCost()
00020     {
00021         return upgradeCost;
00022     }
00023
00024     public void ApplyUpgrade(BaseEntity player)
00025     {
00026         Debug.Log("Upgrade " + upgradeName + " applied to " + player.GetEntityName());
00027     }
00028 }
```

7.20 CameraFollow.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 public class CameraFollow : MonoBehaviour
00006 {
00007     public Transform player;
00008     [SerializeField] private int cameraSpeed;
00009     private Vector3 _playerVector;
00010
00011
00012     void LateUpdate()
00013     {
00014         if (player != null)
00015         {
00016             _playerVector = player.position;
00017             _playerVector.z = -2;
00018             transform.position = Vector3.Lerp(transform.position, _playerVector, cameraSpeed * Time.deltaTime);
00019         }
00020
00021     }
00022 }
```

7.21 Drunk.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 public class Drunk : BaseEffect
00006 {
00007     [Tooltip("Пират опустошает бутылку рома, на время опьянения пират получает на n меньше урона, но + n% к
      шансу промахнуться")]
00008     [SerializeField] public int additionalDefense;
00009
00010     public override void ApplyEffect(GameObject entity)
00011     {
00012         base.ApplyEffect(entity);
00013         entity.gameObject.GetComponent<MonoBehaviour>().StartCoroutine(DrunkEffect(entity));
00014     }
00015
00016     public IEnumerator DrunkEffect(GameObject entity){
00017         entity.GetComponent<BaseEntity>().IncreaseDefense(additionalDefense);
00018         yield return new WaitForSeconds(duration);
00019         entity.GetComponent<BaseEntity>().DecreaseDefense(additionalDefense);
00020     }
00021 }
```

7.22 OnFire.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 public class OnFire : BaseEffect
00006 {
00007     [SerializeField] private int fireDamage = 5;
00008     [SerializeField] private float damageTickSeconds = 1.0f;
00009     [SerializeField] private int repeatCount = 5;
00010
00011     public override void ApplyEffect(GameObject entity)
00012     {
00013         base.ApplyEffect(entity);
00014         StartCoroutine(OnFireEffect(entity));
00015
00016     }
00017
00018     public IEnumerator OnFireEffect(GameObject entity){
00019         for(int i = 0; i < repeatCount; i++)
00020         {
00021             entity.GetComponent<BaseEntity>().TakeDamage(fireDamage);
00022             yield return new WaitForSeconds(damageTickSeconds);
00023         }
00024     }
00025 }
```

7.23 Poison.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 using Photon.Pun;
00005
00006 public class Poison : BaseEffect
00007 {
00008     [SerializeField] public int posionDamage = 5;
00009     [SerializeField] public float damageTickSeconds = 1.0f;
00010
00011     public override void ApplyEffect(GameObject entity)
00012     {
00013         base.ApplyEffect(entity);
00014         entity.gameObject.GetComponent<MonoBehaviour>().StartCoroutine(PoisonEffect(entity));
00015     }
00016
00017     public IEnumerator PoisonEffect(GameObject entity){
00018         for(int i = 0; i < duration/damageTickSeconds; i++)
00019         {
00020             Debug.Log("Poisoned " + entity.name);
00021             entity.GetComponent<PhotonView>().RPC("TakeDamageRemote", RpcTarget.All,
        entity.GetComponent<PhotonView>().ViewID, posionDamage);
00022             yield return new WaitForSeconds(damageTickSeconds);
00023         }
00024     }
00025 }
```

7.24 SpeedUp.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 public class SpeedUp : BaseEffect
00006 {
00007     [Tooltip("Усокрение")]
00008     [SerializeField] public int additionalSpeed = 2;
00009
00010     public override void ApplyEffect(GameObject entity)
00011     {
00012         base.ApplyEffect(entity);
00013         entity.gameObject.GetComponent<MonoBehaviour>().StartCoroutine(SpeedEffect(entity));
00014     }
00015
00016     public IEnumerator SpeedEffect(GameObject entity){
00017         entity.GetComponent<BaseEntity>().IncreaseSpeed(additionalSpeed);
00018         yield return new WaitForSeconds(duration);
00019         entity.GetComponent<BaseEntity>().DecreaseSpeed(additionalSpeed);
00020     }
00021 }
```

7.25 Strong.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 public class Strong : BaseEffect
00006 {
00007     [Tooltip("Усиление")]
00008     [SerializeField] public int additionalDamage = 30;
00009
00010     public override void ApplyEffect(GameObject entity)
00011     {
00012         base.ApplyEffect(entity);
00013         entity.gameObject.GetComponent<MonoBehaviour>().StartCoroutine(StrongEffect(entity));
00014     }
00015
00016     public IEnumerator StrongEffect(GameObject entity){
00017         entity.GetComponent<BaseEntity>().IncreaseDamage(additionalDamage);
00018         yield return new WaitForSeconds(duration);
00019         entity.GetComponent<BaseEntity>().DecreaseDamage(additionalDamage);
00020     }
00021 }
```

7.26 EndGame.cs

```
00001 using System;
00002 using UnityEngine;
00006 public class EndGame : MonoBehaviour
00007 {
00008     public static Action<string> OnGameEnd; // Событие окончания игры
00009
00010     private void Start()
00011     {
00012         Debug.Log("Waiting end game");
00013     }
00014
00015     public void TeamOneWin()
00016     {
00017         Debug.Log("TEAM ONE WIN");
00018         OnGameEnd?.Invoke("TeamOne");
00019     }
00020     public void TeamTwoWin()
00021     {
00022         Debug.Log("TEAM TWO WIN");
00023         OnGameEnd?.Invoke("TeamTwo");
00024     }
00025 }
```

7.27 DisplayedItem.cs

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using UnityEngine;
00005 using UnityEngine.UI;
00006 using UnityEngine.EventSystems;
00010 public class DisplayedItem : EventTrigger
00011 {
00012     public RectTransform inventoryPanel; // Целевая панель инвентаря
00013     public RectTransform displayedItem; // Целевая панель предмета
00014     public int itemId; // ID предмета
00015     public Vector2 itemPosition; // сохраненная позиция предмета на экране
00016
00017     public static Action<int> onItemDrop; // Событие выбрасывания предмета из инвентаря
00018     public static Action<DisplayedItem> onItemSwap; // Событие изменения позиции предмета в инвентаре
00019
00020     private bool isDragging; // Индикатор изменения позиции предмета на экране
00021
00022
00023
00024     // Start is called before the first frame update
00025     void Start()
00026     {
00027         itemPosition = transform.position;
00028     }
00029
00030     // Update is called once per frame
00031     void Update()
```

```
00032     {
00033         if (isDragging)
00034         {
00035             transform.position = new Vector2(Input.mousePosition.x, Input.mousePosition.y);
00036         }
00037     }
00042     public override void OnPointerDown(PointerEventData eventData)
00043     {
00044         itemPosition = displayedItem.position;
00045         isDragging = true;
00046     }
00051     public override void OnPointerUp(PointerEventData eventData)
00052     {
00053         isDragging = false;
00054         if (!RectTransformUtility.RectangleContainsScreenPoint(inventoryPanel, Input.mousePosition))
00055         {
00056             onItemDrop?.Invoke(itemId);
00057         }
00058         else
00059         {
00060             onItemSwap?.Invoke(this);
00061         }
00062     }
00063
00064 }
```

7.28 Inventory.cs

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using UnityEngine;
00005 using Photon.Pun;
00006
00007 public class Inventory : MonoBehaviour
00008 {
00009     public Action<BaseItem> onItemAdded;
00010
00011     [SerializeField] public List<BaseItem> inventoryItems = new List<BaseItem>();
00012     [SerializeField] private GameObject _droppedItemPrefab;
00013
00014     PhotonView _view;
00015
00016     private void Start()
00017     {
00018         _view = GetComponent<PhotonView>();
00019     }
00020
00021     public void AddItem(BaseItem item, GameObject itemGameObject){
00022         inventoryItems.Add(item);
00023         onItemAdded?.Invoke(item);
00024         Debug.Log("Added item: " + item.GetItemName() + item.GetDescription() +  item.GetAmount());
00025         Destroy(itemGameObject);
00026     }
00027
00028     public List<BaseItem> GetAllItems(){
00029         return inventoryItems;
00030     }
00031     public void ClearInventory()
00032     {
00033         inventoryItems.Clear();
00034     }
00035
00036
00037     [PunRPC]
00038     public void SyncronisedDrop(int index)
00039     {
00040         var item = inventoryItems[index];
00041         item.SetAmount(item.GetAmount() - 1);
00042
00043         Vector2 position = new Vector2(gameObject.transform.position.x, gameObject.transform.position.y - 1f);
00044         GameObject droppedItem = Instantiate(_droppedItemPrefab, position, Quaternion.identity);
00045         droppedItem.GetComponent<DroppedItem>().SetItem(item);
00046     }
00047
00048     public bool DropItem(int itemId)
00049     {
00050         _view.RPC("SyncronisedDrop", RpcTarget.All, itemId);
00051
00052         var item = inventoryItems[itemId];
00053         if (item.GetAmount() < 1)
00054         {
00055             return false;
00056         }
```

```
00057
00058         return true;
00059     }
00060 }
```

7.29 InventoryWindow.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using UnityEngine;
00005 using UnityEngine.UI;
00006 using UnityEngine.EventSystems;
00007 using System;
00011 public class InventoryWindow : MonoBehaviour
00012 {
00013     private GameObject player; // Объект целевой сущности
00014     [SerializeField] Inventory playerInventory; // Модуль инвентаря целевой сущности
00015     [SerializeField] RectTransform inventoryPanel; // Целевая панель инвентаря
00016
00017
00018
00019     List<GameObject> displayedIcons = new List<GameObject>(); // Список отображаемых предметов
00020
00021     public void Start()
00022     {
00023         if (playerInventory != null) playerInventory.onItemAdded += OnItemAdded;
00024         DisplayedItem.onItemDrop += OnItemDropped;
00025         DisplayedItem.onItemSwap += onItemSwapped;
00026         KeyHandler.keyPressed += KeyPressed;
00027         SpawnPlayers.OnSpawn += OnSpawn;
00028     }
00029     private void OnDestroy()
00030     {
00031         DisplayedItem.onItemDrop -= OnItemDropped;
00032         DisplayedItem.onItemSwap -= onItemSwapped;
00033         KeyHandler.keyPressed -= KeyPressed;
00034         SpawnPlayers.OnSpawn -= OnSpawn;
00035         if (playerInventory != null) playerInventory.onItemAdded -= OnItemAdded;
00036     }
00037
00038     public void OnSpawn(GameObject playerObject)
00039     {
00040         player = playerObject;
00041         playerInventory = playerObject.GetComponent<Inventory>();
00042         playerInventory.onItemAdded += OnItemAdded;
00043         Redraw();
00044     }
00045
00046     private void Update()
00047     {
00048
00049     }
00050
00051     void onItemSwapped(DisplayedItem item) => OnItemSwap(item);
00052     void OnItemDropped(int itemId) => OnItemDrop(itemId);
00053     void OnItemAdded(BaseItem item) => Redraw();
00059     void KeyPressed(string name, KeyCode key)
00060     {
00061         string[] words = name.Split(' ');
00062         if (words[0] != "Slot") return;
00063         int num = Convert.ToInt32(words[1]) - 1;
00064         if (displayedIcons.Count() > num)
00065         {
00066             bool stillHas = playerInventory.inventoryItems[num].UseItem(player);
00067             if (!stillHas)
00068             {
00069                 playerInventory.inventoryItems.RemoveAt(num);
00070             }
00071             Redraw();
00072         }
00073     }
00078     void OnItemSwap(DisplayedItem item)
00079     {
00080         bool swapped = false;
00081         foreach (GameObject icon in displayedIcons)
00082         {
00083             RectTransform iconTransform = icon.GetComponent<RectTransform>();
00084             Vector2 mousePos = Input.mousePosition;
00085             if (RectTransformUtility.RectangleContainsScreenPoint(iconTransform, mousePos))
00086             {
00087                 int newId = icon.GetComponent<DisplayedItem>().itemId;
00088                 int oldId = item.itemId;
00089                 if (newId == oldId) continue;
```

```
00090                 BaseItem swapping = playerInventory.inventoryItems[newId];
00091                 playerInventory.inventoryItems[newId] = playerInventory.inventoryItems[oldId];
00092                 playerInventory.inventoryItems[oldId] = swapping;
00093                 swapped = true;
00094                 break;
00095             }
00096         }
00097         if (swapped) Redraw();
00098         else
00099         {
00100             item.transform.position = item.itemPosition;
00101         }
00102     }
00107     void OnItemDrop(int itemId)
00108     {
00109         if (displayedIcons.Count > itemId && itemId >= 0)
00110         {
00111             if (!playerInventory.DropItem(itemId))
00112             {
00113                 playerInventory.inventoryItems.RemoveAt(itemId);
00114             }
00115             Redraw();
00116         }
00117     }
00121     void Redraw()
00122     {
00123         ClearDisplayedItems();
00124         for (var i = 0; i < playerInventory.inventoryItems.Count; i++)
00125         {
00126             var item = playerInventory.inventoryItems[i];
00127 
00128             if (item != null)
00129             {
00130                 var icon = new GameObject(name: "Icon");
00131                 icon.AddComponent<Image>().sprite = item.GetImage();
00132                 icon.AddComponent<DisplayedItem>().displayedItem = icon.GetComponent<RectTransform>();
00133                 icon.GetComponent<DisplayedItem>().inventoryPanel = inventoryPanel;
00134                 icon.GetComponent<DisplayedItem>().itemId = i;
00135                 icon.AddComponent<TooltipTextUI>().text = "\"" + item.GetItemName() + "\"\n" +
    item.GetDescription() + "\n" + "Amount: " + item.GetAmount();
00136                 icon.transform.SetParent(inventoryPanel);
00137                 displayedIcons.Add(icon);
00138             }
00139         }
00140     }
00141 
00142     void ClearDisplayedItems()
00143     {
00144         for (var i = 0; i < displayedIcons.Count; i++)
00145         {
00146             Destroy(displayedIcons[i]);
00147         }
00148         displayedIcons.Clear();
00149     }
00150 }
```

7.30 Appe.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 using Photon.Pun;
00005 
00006 
00007 [CreateAssetMenu(fileName = "Apple", menuName = "ScriptableObject/Items/Apple")]
00008 public class Appe : BaseItem
00009 {
00010     public override bool UseItem(GameObject owner)
00011     {
00012         var _view = owner.GetComponent<PhotonView>();
00013         owner.GetComponent<BaseEntity>().GetHealthPoints();
00014         _view.RPC("Heal", RpcTarget.All, 100.0f);
00015         return base.UseItem(owner);
00016     }
00017 
00018 }
```

7.31 DroppedItem.cs

```
00001 using UnityEngine;
```

```
00002
00003 public class DroppedItem : MonoBehaviour
00004 {
00005     public BaseItem itemData;
00006
00007     private SpriteRenderer _spriteRenderer;
00008     void Start()
00009     {
00010         _spriteRenderer = GetComponent<SpriteRenderer>();
00011         _spriteRenderer.sprite = itemData.GetImage();
00012     }
00013
00014     public void SetItem(BaseItem aitemData)
00015     {
00016         _spriteRenderer = GetComponent<SpriteRenderer>();
00017         this.itemData = aitemData;
00018         _spriteRenderer.sprite = aitemData.GetImage();
00019     }
00020     void OnTriggerEnter2D(Collider2D entity)
00021     {
00022         if (entity.gameObject.CompareTag("Player"))
00023         {
00024             if (entity.gameObject.GetComponent<Inventory>().inventoryItems.Count >= 9) return;
00025             entity.gameObject.GetComponent<Inventory>().AddItem(itemData, gameObject);
00026             Destroy(gameObject);
00027         }
00028     }
00029 }
```

7.32 ManagerGame.cs

```
00001 using System.Collections.Generic;
00002 using GameSettings;
00003 using UnityEngine;
00004 using Photon.Pun;
00005 using Photon.Pun.UtilityScripts;
00006 using Photon.Realtime;
00007 using Hashtable = ExitGames.Client.Photon.Hashtable;
00008
00009 public class ManagerGame : MonoBehaviourPunCallbacks
00010 {
00011     private GameObject[] _playerGameObjectsList;
00012     private bool _allPlayersWasSpawnedInGame;
00013
00014     #region Unity Methods
00015
00016     private void Start()
00017     {
00018         if (PhotonNetwork.IsMasterClient)
00019         {
00020             PhotonNetwork.CurrentRoom.IsOpen = false;
00021             PhotonNetwork.CurrentRoom.IsVisible = false;
00022         }
00023         _allPlayersWasSpawnedInGame = false;
00024         gameObject.GetComponent<SpawnPlayers>().enabled = true;
00025         DistributionByTeams();
00026     }
00027
00028     private void Update()
00029     {
00030         if (!_allPlayersWasSpawnedInGame)
00031         {
00032             AllPlayersWasSpawned();
00033         }
00034     }
00035
00036     #endregion
00037
00038     #region Private Methods
00039
00043     private void AllPlayersWasSpawned()
00044     {
00045         _playerGameObjectsList = GameObject.FindGameObjectsWithTag("Player");
00046         if (_playerGameObjectsList.Length == GameSettingsOriginal.MaxPlayersInGame)
00047         {
00048             _allPlayersWasSpawnedInGame = true;
00049             WhenAllPlayersSpawned();
00050         }
00051     }
00052
00056     private void WhenAllPlayersSpawned()
00057     {
00058         gameObject.GetComponent<EndGame>().enabled = true;
00059     }
```

```
00060
00064     private void DistributionByTeams()
00065     {
00066         if (PhotonNetwork.IsMasterClient)
00067         {
00068             PhotonTeamsManager teams = gameObject.GetComponent<PhotonTeamsManager>();
00069             var teamNames = new List<string>();
00070             foreach (var p in PhotonNetwork.PlayerList)
00071             {
00072                 var pTeam = p.CustomProperties["team"].ToString();
00073                 if (!teamNames.Contains(pTeam) && pTeam != "None")
00074                 {
00075                     teamNames.Add(pTeam);
00076                 }
00077             }
00078
00079             if (teamNames.Count == 0)
00080             {
00081                 var count = 0;
00082                 foreach (var player in PhotonNetwork.PlayerList)
00083                 {
00084                     if (player.GetPhotonTeam() != null) player.LeaveCurrentTeam();
00085                     if (count < GameSettingsOriginal.MaxPlayersInGame / 2)
00086                     {
00087                         player.JoinTeam("TeamOne");
00088                         count++;
00089                     }
00090                     else
00091                     {
00092                         player.JoinTeam("TeamTwo");
00093                     }
00094                 }
00095             }
00096             else if (teamNames.Count == 1)
00097             {
00098                 foreach (var player in PhotonNetwork.PlayerList)
00099                 {
00100                     var playerTeam = player.CustomProperties["team"].ToString();
00101                     if (playerTeam != "None")
00102                     {
00103                         player.JoinTeam("TeamOne");
00104                     }
00105                     else
00106                     {
00107                         player.JoinTeam("TeamTwo");
00108                     }
00109                 }
00110             }
00111             else if (teamNames.Count == 2)
00112             {
00113                 foreach (var player in PhotonNetwork.PlayerList)
00114                 {
00115                     var playerTeam = player.CustomProperties["team"].ToString();
00116                     if (playerTeam == teamNames[0])
00117                     {
00118                         player.JoinTeam("TeamOne");
00119                     }
00120                     else
00121                     {
00122                         player.JoinTeam("TeamTwo");
00123                     }
00124                 }
00125             }
00126         }
00127     }
00128
00134     private int GetTeamMembersNum(string teamName)
00135     {
00136         Player[] players;
00137         if (PhotonTeamsManager.Instance.TryGetTeamMembers(teamName, out players))
00138         {
00139             return players.Length;
00140         }
00141
00142         return 0;
00143     }
00144
00145     #endregion
00146
00147     #region Public Methods
00148
00154     public override void OnPlayerPropertiesUpdate(Player targetPlayer, Hashtable changedProps)
00155     {
00156         gameObject.GetComponent<SpawnEnemy>().enabled = true;
00157         //gameObject.GetComponent<SpawnPlayers>().enabled = true;
00158     }
00159
```

```
00160     #endregion
00161 }
```

7.33 Generator.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using UnityEngine;
00005 using UnityEngine.SceneManagement;
00006
00007 public class Generator : MonoBehaviour
00008 {
00009     public Room[] RoomPrefabs;
00010     public Room StartingRoom;
00011     public int Width;
00012     public int Height;
00013
00014     private Room[,] spawnedRooms;
00015     //IEnumerator
00016     private void Start()
00017     {
00018         spawnedRooms = new Room[Width, Height];
00019         spawnedRooms[Width / 2, 0] = StartingRoom;
00020         Debug.Log("" +Width / 2 + " " + 0);
00021
00022
00023         for (int i = 0; i < Width * Height - 1; i++)
00024         {
00025             //yield return new WaitForSecondsRealtime(0.5f);
00026             PlaceOneRoom();
00027         }
00028     }
00029
00030     private void PlaceOneRoom()
00031     {
00032         HashSet<Vector2Int> vacantPlaces = new HashSet<Vector2Int>();
00033         for (int x = 0; x < spawnedRooms.GetLength(0); x++)
00034         {
00035             for (int y = 0; y < spawnedRooms.GetLength(1); y++)
00036             {
00037                 if (spawnedRooms[x, y] == null) continue;
00038
00039                 int maxX = spawnedRooms.GetLength(0) - 1;
00040                 int maxY = spawnedRooms.GetLength(1) - 1;
00041
00042                 if (x > 0 && spawnedRooms[x - 1, y] == null) vacantPlaces.Add(new Vector2Int(x - 1, y));
00043                 if (y > 0 && spawnedRooms[x, y - 1] == null) vacantPlaces.Add(new Vector2Int(x, y - 1));
00044                 if (x < maxX && spawnedRooms[x + 1, y] == null) vacantPlaces.Add(new Vector2Int(x + 1, y));
00045                 if (y < maxY && spawnedRooms[x, y + 1] == null) vacantPlaces.Add(new Vector2Int(x, y + 1));
00046             }
00047         }
00048         Room newRoom = Instantiate(RoomPrefabs[Random.Range(0, RoomPrefabs.Length)]);
00049
00050         Vector2Int position = vacantPlaces.ElementAt(Random.Range(0, vacantPlaces.Count));
00051         newRoom.transform.position = new Vector3(position.x +1, position.y+1, 0) * Random.Range(12, 20);
00052         spawnedRooms[position.x, position.y] = newRoom;
00053         Debug.Log(""+ position.x+" " + position.y);
00054
00055
00056     }
00057
00058     //private bool ConnectToSomething(Room room, Vector2Int p)
00059     //{
00060     //    int maxX = spawnedRooms.GetLength(0) - 1;
00061     //    int maxY = spawnedRooms.GetLength(1) - 1;
00062
00063     //    List<Vector2Int> neighbours = new List<Vector2Int>();
00064
00065     //    if (room.DoorU != null && p.y < maxY && spawnedRooms[p.x, p.y + 1]?.DoorD != null)
        neighbours.Add(Vector2Int.up);
00066     //    if (room.DoorD != null && p.y > 0 && spawnedRooms[p.x, p.y - 1]?.DoorU != null)
        neighbours.Add(Vector2Int.down);
00067     //    if (room.DoorR != null && p.x < maxX && spawnedRooms[p.x + 1, p.y]?.DoorL != null)
        neighbours.Add(Vector2Int.right);
00068     //    if (room.DoorL != null && p.x > 0 && spawnedRooms[p.x - 1, p.y]?.DoorR != null)
        neighbours.Add(Vector2Int.left);
00069
00070     //    if (neighbours.Count == 0) return false;
00071
00072     //    Vector2Int selectedDirection = neighbours[Random.Range(0, neighbours.Count)];
00073     //    Room selectedRoom = spawnedRooms[p.x + selectedDirection.x, p.y + selectedDirection.y];
00074
00075     //    if (selectedDirection == Vector2Int.up)
```

```
00076     //    {
00077     //        room.DoorU.SetActive(false);
00078     //        selectedRoom.DoorD.SetActive(false);
00079     //    }
00080     //    else if (selectedDirection == Vector2Int.down)
00081     //    {
00082     //        room.DoorD.SetActive(false);
00083     //        selectedRoom.DoorU.SetActive(false);
00084     //    }
00085     //    else if (selectedDirection == Vector2Int.right)
00086     //    {
00087     //        room.DoorR.SetActive(false);
00088     //        selectedRoom.DoorL.SetActive(false);
00089     //    }
00090     //    else if (selectedDirection == Vector2Int.left)
00091     //    {
00092     //        room.DoorL.SetActive(false);
00093     //        selectedRoom.DoorR.SetActive(false);
00094     //    }
00095
00096     //    return true;
00097     //}
00098 }
```

7.34 Room.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 public class Room : MonoBehaviour
00006 {
00007     // Start is called before the first frame update
00008     void Start()
00009     {
00010
00011     }
00012
00013     public void RotateRandomly()
00014     {
00015         int count = Random.Range(0, 4);
00016
00017         for (int i = 0; i < count; i++)
00018         {
00019             transform.Rotate(0, 90, 0);
00020
00021         }
00022     }
00023 }
```

7.35 StartEndZone.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 public class StartEndZone : MonoBehaviour
00006 {
00007
00008     [SerializeField] private BaseItem item;
00009     private void OnTriggerEnter(Collider player)
00010     {
00011         List<BaseItem> inventory = player.GetComponent<Inventory>().GetComponent<List<BaseItem»();
00012         if (inventory.Contains(item))
00013         {
00014         }
00015     }
00016 }
```

7.36 Berserk.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 public class Berserk : BasePassiveSkill
```

```
00006 {
00007     [Tooltip("Пассивный скилл Берсерк Если уровень ХП пирата ниже n% то пират наносить на n% больше урона")]
00008     [SerializeField] private int additionalDamage;
00009 
00010     [Tooltip("n% (0-100)")]
00011     [SerializeField] private float healthPercent;
00012 
00013     private BaseEntity _casterEntity;
00014 
00015     private string _previousState = "upper";
00016     private string _healthState = "upper";
00017 
00018     private void Start() {
00019         this._casterEntity = gameObject.GetComponent<BaseEntity>();
00020     }
00021 
00022     // Update is called once per frame
00023     void Update()
00024     {
00025         if((float)_casterEntity.GetHealthPoints() < (float)_casterEntity.GetMaxHealthPoints() *
      (float)((float)healthPercent / 100))
00026             _healthState = "lower";
00027         if((float)_casterEntity.GetHealthPoints() >= (float)_casterEntity.GetMaxHealthPoints() *
      (float)((float)healthPercent / 100))
00028             _healthState = "upper";
00029 
00030         if(_previousState != _healthState)
00031         {
00032             _previousState = _healthState;
00033             ChangeDamage();
00034         }
00035     }
00036 
00037     void ChangeDamage()
00038     {
00039         if(_healthState == "lower")
00040             _casterEntity.IncreaseDamage(additionalDamage);
00041         if(_healthState == "upper")
00042             _casterEntity.DecreaseDamage(additionalDamage);
00043     }
00044 }
```

7.37 AnimationPlayer.cs

```
00001 using System;
00002 using UnityEngine;
00003 using Photon.Pun;
00004 using System.Collections;
00005 
00006 
00007 public class AnimationPlayer : MonoBehaviour
00008 {
00009     private PhotonView _view;
00010     private Animator _animator;
00011     private string _currentAnimation = "";
00012 
00013     private AudioSource stepSound;
00014     private bool isStepPlaying;
00015 
00016     private float y;
00017     private float x;
00018     private bool isMoving;
00019     public string movingState;
00020 
00021     public readonly string _attack = "attack_";
00022     public readonly string _move = "move_";
00023     public readonly string _drink = "drink_";
00024     public readonly string _idle = "idle_";
00025     public readonly string _shot = "shot_";
00026 
00027     public readonly string _left = "left";
00028     public readonly string _right = "right";
00029     public readonly string _up = "back";
00030     public readonly string _down = "front";
00031 
00032 
00033     private void Start()
00034     {
00035         _view = GetComponent<PhotonView>();
00036         isStepPlaying = false;
00037         isMoving = false;
00038         _animator = GetComponent<Animator>();
00039         stepSound = GetComponent<AudioSource>();
00040     }
```

```
00041     // _view.RPC("ChangePlayerAnimation", RpcTarget.All, FrontPlayerAnimation);
00042
00043
00044     private void Update()
00045     {
00046         if (_view.IsMine && GetComponent<MovementPlayer>().canMove)
00047         {
00048             getCurrentMovingState();
00049             if (!isMoving)
00050             {
00051                 ChangePlayerAnimation_q(_idle);
00052                 //Debug.Log(_idle + movingState + "nemove");
00053             }
00054             else
00055             {
00056                 ChangePlayerAnimation_q(_move);
00057                 //Debug.Log(_move + movingState+"mofe");
00058                 playStep();
00059             }
00060         }
00061         isMoving = false;
00062     }
00063     public void ChangePlayerAnimation_q(string newAnim)
00064     {
00065         if(!_animator.GetCurrentAnimatorStateInfo(-1).IsName(_attack+movingState))
00066             _view.RPC("ChangePlayerAnimation", RpcTarget.All, newAnim + movingState);
00067     }
00068     public string getCurrentMovingState()
00069     {
00070         y = Input.GetAxisRaw("Vertical");
00071         x = Input.GetAxisRaw("Horizontal");
00072
00073         if (x != 0 || y != 0)
00074         {
00075             isMoving = true;
00076             if (y > 0)
00077                 movingState = _up;
00078             else
00079                 movingState = _down;
00080             if (x > 0)
00081                 movingState = _right;
00082             else if (x != 0)
00083                 movingState = _left;
00084         }
00085         return movingState;
00086     }
00087
00088     void playStep()
00089     {
00090         if (!isStepPlaying)
00091         {
00092             stepSound.Play();
00093             stepSound.pitch = new System.Random().Next(80, 100) / 100;
00094             StartCoroutine(stepDelay(0.84f));
00095         }
00096     }
00097
00098     private IEnumerator stepDelay(float delay)
00099     {
00100         isStepPlaying=true;
00101         yield return new WaitForSeconds(delay);
00102         isStepPlaying = false;
00103     }
00104     [PunRPC]
00105     public void ChangePlayerAnimation(string newAnimation)
00106     {
00107         if (_currentAnimation == newAnimation) return;
00108         _currentAnimation = newAnimation;
00109         //Debug.Log(newAnimation+"Photon");
00110         _animator.Play(_currentAnimation);
00111     }
00112 }
```

7.38 CameraPlayer.cs

```
00001 using UnityEngine;
00002 using Photon.Pun;
00003
00004 public class CameraPlayer : MonoBehaviour
00005 {
00006     private PhotonView _view;
00007     private void Start()
00008     {
00009         _view = GetComponent<PhotonView>();
```

```
00010         if (_view.Owner.IsLocal)
00011         {
00012             Camera.main.GetComponent<CameraFollow>().player = gameObject.transform;
00013         }
00014     }
00015 }
```

7.39 ClassSelection.cs

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using UnityEngine;
00005 using UnityEngine.UI;
00006 using Photon.Pun;
00007 using Photon.Pun.UtilityScripts;
00011 public class ClassSelection : MonoBehaviour
00012 {
00013     bool classSet = false; // Индикатор выбора класса
00014 
00015     [SerializeField, Tooltip("Высота каждой кнопки выбора клавча")] int height;
00016     [SerializeField, Tooltip("Объект панели выбора класса")] GameObject classSelection;
00017     [SerializeField, Tooltip("Префаб для кнопки выбора класса/подкласса")] GameObject SubclassButton;
00018     [SerializeField, Tooltip("Модуль появления игрока")] SpawnPlayers spawnPlayers;
00019     #region UI Panels
00020     [SerializeField, Tooltip("Поле названия первого класса")] private Text firstClassName;
00021     [SerializeField, Tooltip("Поле названия второго класса")] private Text secondClassName;
00022     [SerializeField, Tooltip("Поле названия третьего класса")] private Text thirdClassName;
00023 
00024     [SerializeField] private RectTransform firstClass;
00025     [SerializeField] private RectTransform secondClass;
00026     [SerializeField] private RectTransform thirdClass;
00027 
00028     [SerializeField, Tooltip("Список объектов (префабов) для подклассов первого класса")] private GameObject[]
      firstClassSubclasses;
00029     [SerializeField, Tooltip("Список объектов (префабов) для подклассов второго класса")] private GameObject[]
      secondClassSubclasses;
00030     [SerializeField, Tooltip("Список объектов (префабов) для подклассов третьего класса")] private GameObject[]
      thirdClassSubclasses;
00031 
00032     [SerializeField, Tooltip("Панель для привязки кнопок выбора подклассов первого класса")] private RectTransform
      firstClassSubclassesPanel;
00033     [SerializeField, Tooltip("Панель для привязки кнопок выбора подклассов второго класса")] private RectTransform
      secondClassSubclassesPanel;
00034     [SerializeField, Tooltip("Панель для привязки кнопок выбора подклассов третьего класса")] private RectTransform
      thirdClassSubclassesPanel;
00035     #endregion
00036     // Start is called before the first frame update
00037     void Start()
00038     {
00039         TimerManager.timerEnd += OnTimerEnd;
00040 
00041         firstClassSubclassesPanel.sizeDelta = new Vector2(356, firstClassSubclasses.Length * height + 50);
00042         secondClassSubclassesPanel.sizeDelta = new Vector2(356, secondClassSubclasses.Length * height + 50);
00043         thirdClassSubclassesPanel.sizeDelta = new Vector2(356, thirdClassSubclasses.Length * height + 50);
00044 
00045         SpawnSubclassButtons(firstClassSubclassesPanel, firstClassSubclasses);
00046         SpawnSubclassButtons(secondClassSubclassesPanel, secondClassSubclasses);
00047         SpawnSubclassButtons(thirdClassSubclassesPanel, thirdClassSubclasses);
00048 
00049     }
00054     void OnTimerEnd(bool ended)
00055     {
00056         if (ended && !classSet)
00057         {
00058             var classNum = UnityEngine.Random.Range(1, 3);
00059             var cls = firstClassSubclasses;
00060             switch (classNum)
00061             {
00062                 case 1:
00063                     cls = firstClassSubclasses;
00064                     break;
00065                 case 2:
00066                     cls = secondClassSubclasses;
00067                     break;
00068                 case 3:
00069                     cls = thirdClassSubclasses;
00070                     break;
00071                 default:
00072                     break;
00073             }
00074 
00075             SetPlayerClass(cls[UnityEngine.Random.Range(0, cls.Length)]);
00076         }
```

```
00077         TimerManager.timerEnd -= OnTimerEnd;
00078         classSelection.SetActive(false);
00079     }
00085     void SpawnSubclassButtons(RectTransform parent, GameObject[] prefabs)
00086     {
00087         int posY = -75;
00088         foreach (GameObject playerPrefab in prefabs)
00089         {
00090             var subclass = Instantiate(SubclassButton, new Vector3(0, 0, 0), Quaternion.identity);
00091             subclass.transform.SetParent(parent.transform, false);
00092             subclass.GetComponent<RectTransform>().sizeDelta = new Vector2(336, height);
00093             subclass.transform.localPosition = new Vector3(0, posY, 0);
00094             subclass.GetComponent<Button>().onClick.AddListener(() => { SetPlayerClass(playerPrefab); });
00095             subclass.GetComponentInChildren<Text>().text =
      playerPrefab.GetComponent<BaseEntity>().GetEntityName();
00096             // subclass.GetComponent<Image>().sprite = playerPrefab.GetComponent<Image>().sprite;
00097             posY -= height + 10;
00098         }
00099     }
00104     void SetPlayerClass(GameObject playerPrefab)
00105     {
00106         if (classSet) return;
00107         firstClass.gameObject.SetActive(false);
00108         secondClass.gameObject.SetActive(false);
00109         thirdClass.gameObject.SetActive(false);
00110         Debug.Log(playerPrefab.GetComponent<BaseEntity>().GetEntityName());
00111         spawnPlayers.SetPlayerObject(playerPrefab);
00112         classSet = true;
00113     }
00114
00115     // Update is called once per frame
00116     void Update()
00117     {
00118     }
00119
00120
00121 }
```

7.40 InputControllerPlayer.cs

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using UnityEngine;
00005
00006 public class InputControllerPlayer : MonoBehaviour
00007 {
00008     private BaseEntity _player;
00009
00010     void Start()
00011     {
00012         KeyHandler.keyPressed += OnKeyPressed;
00013         _player = this.gameObject.GetComponent<BaseEntity>();
00014     }
00015
00016     private void OnDestroy()
00017     {
00018         KeyHandler.keyPressed -= OnKeyPressed;
00019     }
00020
00021     void OnKeyPressed(string name, KeyCode code)
00022     {
00023         //if (name == "Use Skill") _player.UseSkill();
00024         //else if (name.StartsWith("Skill")) _player.selectSkill(name);
00025     }
00026
00027     void Update()
00028     {
00029         /*
00030         if (Input.GetMouseButtonDown(0))
00031         {
00032             Debug.Log("Input");
00033             _player.UseSkill(0);
00034         }
00035         */
00036     }
00037 }
```

7.41 MovementPlayer.cs

```
00001 using Photon.Pun;
```

```
00002 using UnityEngine;
00003 using System.Collections;
00004
00005 public class MovementPlayer : MonoBehaviour
00006 {
00007     private PhotonView _view;
00008     private BaseEntity _player;
00009     private float _moveSpeed;
00010     private Rigidbody2D _rb2d;
00011
00012     public bool canMove = true;
00013
00014     #region KeybindMovement
00015     private bool keybindMovement = false;
00016     #endregion
00017
00018
00019     private void Start()
00020     {
00021         _player = GetComponent<BaseEntity>();
00022         _moveSpeed = _player.GetMoveSpeed();
00023         _rb2d = GetComponent<Rigidbody2D>();
00024         _view = GetComponent<PhotonView>();
00025         SettingsManager.keybindMovementToggled += ToggleKeybindMovement;
00026     }
00027     private void OnDestroy()
00028     {
00029         SettingsManager.keybindMovementToggled -= ToggleKeybindMovement;
00030     }
00031     void ToggleKeybindMovement(bool isOn)
00032     {
00033         keybindMovement = isOn;
00034     }
00035
00036     public void cancelMovement(float stopTime)
00037     {
00038         StartCoroutine(movementStopCoroutine(stopTime));
00039     }
00040
00041     private IEnumerator movementStopCoroutine(float stopTime)
00042     {
00043         canMove = false;
00044         yield return new WaitForSeconds(stopTime);
00045         canMove = true;
00046     }
00047
00048     private void FixedUpdate()
00049     {
00050         _moveSpeed = _player.GetMoveSpeed();
00051         if(!canMove) return;
00052         var keyHandler = KeyHandler.instance;
00053         if (keyHandler.IsPaused()) return;
00054
00055         if (_view.IsMine)
00056         {
00057             if (!keybindMovement)
00058             {
00059                 Vector2 movement = new Vector2(Input.GetAxisRaw("Horizontal"), Input.GetAxisRaw("Vertical"));
00060                 _rb2d.MovePosition(_rb2d.position + movement * _moveSpeed * Time.fixedDeltaTime);
00061             }
00062             else
00063             {
00064                 int velX = 0, velY = 0;
00065                 if (Input.GetKey(keyHandler.GetKeybind("Right")))
00066                 {
00067                     velX += 1;
00068                 }
00069                 if (Input.GetKey(keyHandler.GetKeybind("Left")))
00070                 {
00071                     velX -= 1;
00072                 }
00073                 if (Input.GetKey(keyHandler.GetKeybind("Forward")))
00074                 {
00075                     velY += 1;
00076                 }
00077                 if (Input.GetKey(keyHandler.GetKeybind("Backward")))
00078                 {
00079                     velY -= 1;
00080                 }
00081                 Vector2 movement = new Vector2(velX, velY);
00082                 _rb2d.MovePosition(_rb2d.position + movement * _moveSpeed * Time.fixedDeltaTime);
00083             }
00084         }
00085     }
00086 }
```

7.42 PlayerScript.cs

```
00001 using Photon.Pun;
00002 using Photon.Pun.UtilityScripts;
00003 using UnityEngine;
00004
00005 public class PlayerScript : MonoBehaviour
00006 {
00007     private BaseEntity _playerEntitiy;
00008
00009     private GameObject _gameManager;
00010     private PlayersTeamsManager _playersTeamsManager;
00011     private PhotonView _view;
00012
00013
00014     private void Start()
00015     {
00016         _view = gameObject.GetComponent<PhotonView>();
00017         _playerEntitiy = gameObject.GetComponent<BaseEntity>();
00018
00019         _gameManager = GameObject.FindGameObjectWithTag("GameManager");
00020         _playersTeamsManager = _gameManager.GetComponent<PlayersTeamsManager>();
00021
00022         if(_view.IsMine)
00023             KeyHandler.keyPressed += OnKeyPressed;
00024     }
00025     private void OnDestroy()
00026     {
00027         if (_view.IsMine) KeyHandler.keyPressed -= OnKeyPressed;
00028     }
00029
00030     private void Update() {
00031         if(_view.IsMine)
00032             _playerEntitiy.TickPoints();
00033     }
00034
00035     // Использование скиллов по кнопке
00036     void OnKeyPressed(string name, KeyCode key)
00037     {
00038         Debug.Log("KeyPressed" + name);
00039         if (name == "Attack")
00040         {
00041             if(_playerEntitiy.GetEntityName() == "Pirate") GetComponent<PlayerMelee>().MasterCheckMeleeAttack();
00042             if(_playerEntitiy.GetEntityName() == "Mage")
      _playerEntitiy.gameObject.GetComponent<PlayerProjectile>().Attack("Fireball", 1.5f);
00043             AnimationPlayer _anim = GetComponent<AnimationPlayer>();
00044             _anim.ChangePlayerAnimation_q(_anim._attack);
00045         }
00046         if (name == "Use Skill") _playerEntitiy.UseSkill();
00047         else if (name.StartsWith("Skill"))
00048         {
00049             var current = _playerEntitiy.GetSelectedSkill();
00050             _playerEntitiy.deSelectSkill();
00051             if (current != name)
00052             {
00053                 _playerEntitiy.selectSkill(name);
00054             }
00055         }
00056     }
00057
00058
00059     public void KillPlayer()
00060     {
00061         var myPlayerView = gameObject.GetComponent<PhotonView>();
00062
00063         Debug.Log("PLAYER DIED FROM: " + myPlayerView.Owner.GetPhotonTeam().Name);
00064         if (myPlayerView.Owner.GetPhotonTeam().Name == "TeamOne")
00065         {
00066             _playersTeamsManager.PlayerInTeamOneDied();
00067         }
00068
00069         if (myPlayerView.Owner.GetPhotonTeam().Name == "TeamTwo")
00070         {
00071             _playersTeamsManager.PlayerInTeamTwoDied();
00072         }
00073         gameObject.SetActive(false);
00074         //PhotonNetwork.Destroy(gameObject);
00075     }
00076 }
```

7.43 PlayerUpgrades.cs

```
00001 using System;
```

```
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using System.Linq;
00005 using UnityEngine;
00006 using UnityEngine.UI;
00010 public class PlayerUpgrades : MonoBehaviour
00011 {
00012     [SerializeField] private BaseEntity player;
00013     [SerializeField] private RectTransform upgradesPanel;
00014     [SerializeField] private int xpPoints;
00015     [SerializeField] private RectTransform classPanel;
00016     [SerializeField] private RectTransform subclassPanel;
00017 
00018     void Start()
00019     {
00020         KeyHandler.keyPressed += KeyPressed;
00021         SpawnPlayers.OnSpawn += OnSpawn;
00022     }
00023     private void OnDestroy()
00024     {
00025         SpawnPlayers.OnSpawn -= OnSpawn;
00026         KeyHandler.keyPressed -= KeyPressed;
00027     }
00028 
00029     void OnSpawn(GameObject playerObject)
00030     {
00031         player = playerObject.GetComponent<BaseEntity>();
00032     }
00033 
00034     void Update()
00035     {
00036     }
00037 
00038     void KeyPressed(string name, KeyCode key)
00039     {
00040         if (name == "Upgrades")
00041         {
00042             upgradesPanel.gameObject.SetActive(!upgradesPanel.gameObject.activeSelf);
00043             KeyHandler.instance.SetUIOpened(upgradesPanel.gameObject.activeSelf);
00044 
00045         }
00046         if (name == "EscapeMenu")
00047         {
00048             if (upgradesPanel?.gameObject.activeSelf == true)
00049                 upgradesPanel.gameObject.SetActive(false);
00050         }
00051     }
00052 
00053     public int GetXp()
00054     {
00055         return xpPoints;
00056     }
00057     public void AddXp(int points)
00058     {
00059         if (points < 0) return;
00060         xpPoints += points;
00061     }
00062     public void AddUpgrade(Button btn)
00063     {
00064         if (xpPoints >= btn.GetComponent<BaseUpgrade>().GetCost())
00065         {
00066             xpPoints -= btn.GetComponent<BaseUpgrade>().GetCost();
00067             btn.onClick.RemoveAllListeners();
00068             Color col = btn.GetComponent<Image>().color;
00069             col.a = 0.1f;
00070             btn.GetComponent<BaseUpgrade>().ApplyUpgrade(player);
00071         }
00072 
00073     }
00074     public void SwitchPanels(bool toClass = true)
00075     {
00076         classPanel.gameObject.SetActive(toClass);
00077         subclassPanel.gameObject.SetActive(!toClass);
00078     }
00079 }
```

7.44 Drunkard.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 
00005 public class Drunkard : BaseSkill
00006 {
```

```
00007     [SerializeField] private GameObject drunkEffect;
00008
00009     [Tooltip("Пират опустошает бутылку рома, на время опьянения пират получает на n меньше урона, но + n% к
      шансу промахнуться")]
00010     [SerializeField] private int additionalDefense = 5;
00011
00012     [Tooltip("Время опьянения (секунды)")]
00013     [SerializeField] private float drunkTime = 10;
00014
00015     public override IEnumerator UseSkill(GameObject caster, string key)
00016     {
00017         base.UseSkill(caster, key);
00018         if (caster.GetComponent<BaseEntity>().spendEnergy(energyCost) == false)
00019         {
00020             Debug.Log("Not enough energy to cast " + this.gameObject.name);
00021             yield break;
00022         }
00023
00024         onCast?.Invoke(castTime);
00025         caster.GetComponent<BaseEntity>().setIsCooldown(key, true);
00026         if(cancelMovementOnCast)
00027             caster.GetComponent<MovementPlayer>().cancelMovement(castTime);
00028
00029         AnimationPlayer anim = caster.GetComponent<AnimationPlayer>();
00030         anim.ChangePlayerAnimation_q(anim._drink);
00031         yield return new WaitForSeconds(castTime);
00032
00033         // Сам скилл
00034         drunkEffect.GetComponent<Drunk>().additionalDefense = additionalDefense;
00035         drunkEffect.GetComponent<Drunk>().duration = drunkTime;
00036         onRelease?.Invoke(cooldown);
00037
00038         caster.GetComponent<BaseEntity>().AddEffect(drunkEffect);
00039         // Сам скилл
00040
00041         yield return new WaitForSeconds(cooldown);
00042         caster.GetComponent<BaseEntity>().setIsCooldown(key, false);
00043     }
00044 }
```

7.45 Exclamation.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 public class Exclamation : BaseSkill
00006 {
00007     [SerializeField] GameObject speedEffect;
00008     [SerializeField] GameObject strongEffect;
00009
00010     public override IEnumerator UseSkill(GameObject caster, string key)
00011     {
00012         base.UseSkill(caster, key);
00013         if (caster.GetComponent<BaseEntity>().spendEnergy(energyCost) == false)
00014         {
00015             Debug.Log("Not enough energy to cast " + this.gameObject.name);
00016             yield break;
00017         }
00018
00019         onCast?.Invoke(castTime);
00020         caster.GetComponent<BaseEntity>().setIsCooldown(key, true);
00021         if(cancelMovementOnCast)
00022             caster.GetComponent<MovementPlayer>().cancelMovement(castTime);
00023         yield return new WaitForSeconds(castTime);
00024
00025         // Сам скилл
00026         caster.GetComponent<BaseEntity>().AddEffect(speedEffect);
00027         caster.GetComponent<BaseEntity>().AddEffect(strongEffect);
00028         // Сам скилл
00029         onRelease?.Invoke(cooldown);
00030         yield return new WaitForSeconds(cooldown);
00031         caster.GetComponent<BaseEntity>().setIsCooldown(key, false);
00032     }
00033 }
```

7.46 FireballSkill.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
```

```
00003 using UnityEngine;
00004 using Photon.Pun;
00005
00006 public class FireballSkill : BaseSkill
00007 {
00008     [SerializeField] private GameObject projectile;
00009
00010     public override IEnumerator UseSkill(GameObject caster, string key)
00011     {
00012         base.UseSkill(caster, key);
00013
00014         if (caster.GetComponent<BaseEntity>().spendEnergy(energyCost) == false)
00015         {
00016             Debug.Log("Not enough energy to cast " + this.gameObject.name);
00017             yield break;
00018         }
00019
00020         onCast?.Invoke(castTime);
00021         caster.GetComponent<BaseEntity>().setIsCooldown(key, true);
00022         if(cancelMovementOnCast)
00023             caster.GetComponent<MovementPlayer>().cancelMovement(castTime);
00024         yield return new WaitForSeconds(castTime);
00025
00026
00027         caster.GetComponent<PlayerProjectile>().Attack("Fireball");
00028         onRelease?.Invoke(cooldown);
00029         yield return new WaitForSeconds(cooldown);
00030         caster.GetComponent<BaseEntity>().setIsCooldown(key, false);
00031     }
00032 }
```

7.47 FlurryOfFire.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 using Photon.Pun;
00005
00006 public class FlurryOfFire : BaseSkill
00007 {
00008     [SerializeField] private GameObject projectile;
00009     [SerializeField] private float shootTime;
00010     [SerializeField] private float timeBetweenShoot;
00011
00012
00013     public override IEnumerator UseSkill(GameObject caster, string key)
00014     {
00015         base.UseSkill(caster, key);
00016         if (caster.GetComponent<BaseEntity>().spendEnergy(energyCost) == false)
00017         {
00018             Debug.Log("Not enough energy to cast " + this.gameObject.name);
00019             yield break;
00020         }
00021
00022         onCast?.Invoke(castTime);
00023         caster.GetComponent<BaseEntity>().setIsCooldown(key, true);
00024         if(cancelMovementOnCast)
00025             caster.GetComponent<MovementPlayer>().cancelMovement(castTime);
00026         yield return new WaitForSeconds(castTime);
00027
00028         // Сам скилл
00029         for(int i = 0; i < shootTime / timeBetweenShoot; i++)
00030         {
00031             caster.GetComponent<PlayerProjectile>().Attack("Bullet");
00032             yield return new WaitForSeconds(timeBetweenShoot);
00033         }
00034         // Сам скилл
00035         onRelease?.Invoke(cooldown);
00036         yield return new WaitForSeconds(cooldown);
00037         caster.GetComponent<BaseEntity>().setIsCooldown(key, false);
00038     }
00039 }
```

7.48 Toxicant.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 using Photon.Pun;
00005
```

```
00006 public class Toxicant : BaseSkill
00007 {
00008     public override IEnumerator UseSkill(GameObject caster, string key)
00009     {
00010         base.UseSkill(caster, key);
00011         if (caster.GetComponent<BaseEntity>().spendEnergy(energyCost) == false)
00012         {
00013             Debug.Log("Not enough energy to cast " + this.gameObject.name);
00014             yield break;
00015         }
00016 
00017         onCast?.Invoke(castTime);
00018         caster.GetComponent<BaseEntity>().setIsCooldown(key, true);
00019         if(cancelMovementOnCast)
00020             caster.GetComponent<MovementPlayer>().cancelMovement(castTime);
00021         yield return new WaitForSeconds(castTime);
00022 
00023         // Сам скилл
00024         caster.GetComponent<PlayerAOE>().Attack("AcidFloor");
00025         // Сам скилл
00026         onRelease?.Invoke(cooldown);
00027         yield return new WaitForSeconds(cooldown);
00028         caster.GetComponent<BaseEntity>().setIsCooldown(key, false);
00029     }
00030 }
```

7.49 SoundManager.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00007 public class SoundManager : MonoBehaviour
00008 {
00009     [SerializeField, Tooltip("Категория звука целевого источника")] string Category; // Категория звука целевого
      источника
00010     [SerializeField, Tooltip("Целевой источник звука")] AudioSource source; // Целевой источник звука
00011     private bool volumeSet = false; // Индикатор установки громкости звука целевого источника
00012     private float volume; // Громкость звука целевого источника
00013     // Start is called before the first frame update
00014     void Start()
00015     {
00016         if (SoundSettings.instance == null) return;
00017         volume = SoundSettings.instance.GetVolume(Category);
00018         source.volume = volume * SoundSettings.instance.GetVolume("General volume");
00019         SoundSettings.volumeChange += VolumeUpdated;
00020         volumeSet = true;
00021     }
00022     private void OnDestroy()
00023     {
00024         SoundSettings.volumeChange -= VolumeUpdated;
00025     }
00026 
00027     private void Update()
00028     {
00029         if (volumeSet || SoundSettings.instance == null) return;
00030         volume = SoundSettings.instance.GetVolume(Category);
00031         source.volume = volume * SoundSettings.instance.GetVolume("General volume");
00032         SoundSettings.volumeChange += VolumeUpdated;
00033         volumeSet = true;
00034     }
00040     void VolumeUpdated(string name, float value)
00041     {
00042         if (name == Category)
00043         {
00044             source.volume = value * SoundSettings.instance.GetVolume("General volume");
00045             volume = value;
00046         }
00047         else if (name == "General volume")
00048         {
00049             source.volume = volume * value;
00050         }
00051     }
00052 }
```

7.50 SpawnEnemy.cs

```
00001 using System.Collections.Generic;
00002 using Photon.Pun;
00003 using UnityEngine;
00004 
```

```
00005 public class SpawnEnemy : MonoBehaviour
00006 {
00007     public GameObject goblinSeekerGameObject2;
00008 
00009     private void Start()
00010     {
00011         if (PhotonNetwork.IsMasterClient)
00012         {
00013             PhotonNetwork.Instantiate(goblinSeekerGameObject2.name, new Vector3(0, 40, 0), Quaternion.identity);
00014             PhotonNetwork.Instantiate(goblinSeekerGameObject2.name, new Vector3(12, 38, 0), Quaternion.identity);
00015             PhotonNetwork.Instantiate(goblinSeekerGameObject2.name, new Vector3(16, 24, 0), Quaternion.identity);
00016             PhotonNetwork.Instantiate(goblinSeekerGameObject2.name, new Vector3(-5, 24, 0), Quaternion.identity);
00017             PhotonNetwork.Instantiate(goblinSeekerGameObject2.name, new Vector3(-12, 13, 0), Quaternion.identity);
00018             PhotonNetwork.Instantiate(goblinSeekerGameObject2.name, new Vector3(-9, -3, 0), Quaternion.identity);
00019             PhotonNetwork.Instantiate(goblinSeekerGameObject2.name, new Vector3(-5, -12, 0), Quaternion.identity);
00020             PhotonNetwork.Instantiate(goblinSeekerGameObject2.name, new Vector3(-6, -12, 0), Quaternion.identity);
00021         }
00022     }
00023 }
```

7.51 SpawnPlayers.cs

```
00001 using System;
00002 using UnityEngine;
00003 using Photon.Pun;
00004 using Photon.Pun.UtilityScripts;
00005 
00006 public class SpawnPlayers : MonoBehaviour
00007 {
00008     private GameObject playerGameObject = null;
00009 
00010     public static Action<GameObject> OnSpawn;
00011 
00012     [SerializeField] private Transform spawnPointTeamOne;
00013     [SerializeField] private Transform spawnPointTeamTwo;
00014     private Vector3 _position;
00015     private bool canSpawn = false;
00016 
00017     private void Start()
00018     {
00019         TimerManager.timerEnd += OnTimerEnd;
00020     }
00021 
00022     void OnTimerEnd(bool ended)
00023     {
00024         canSpawn = ended;
00025     }
00026     private void Update()
00027     {
00028         if (playerGameObject != null && canSpawn)
00029         {
00030             if (PhotonNetwork.LocalPlayer.GetPhotonTeam().Name == "TeamOne")
00031             {
00032                 _position = spawnPointTeamOne.position;
00033             }
00034             else if (PhotonNetwork.LocalPlayer.GetPhotonTeam().Name == "TeamTwo")
00035             {
00036                 _position = spawnPointTeamTwo.position;
00037             }
00038             GameObject playerObject = PhotonNetwork.Instantiate(playerGameObject.name, _position,
      Quaternion.identity);
00039             playerObject.GetComponent<BaseEntity>()._controller = gameObject.GetComponent<Controllers>();
00040             OnSpawn?.Invoke(playerObject);
00041         }
00042         canSpawn = false;
00043     }
00044 
00045     public void SetPlayerObject(GameObject prefab)
00046     {
00047         playerGameObject = prefab;
00048     }
00049 }
```

7.52 PlayersTeamsManager.cs

```
00001 using System;
00002 using UnityEngine;
00003 using GameSettings;
00004 
00005 public class PlayersTeamsManager : MonoBehaviour
```

```
00006 {
00007     private int _lifePlayersInTeamOne;
00008     private int _lifePlayersInTeamTwo;
00009     private EndGame _endGame;
00010     private void Start()
00011     {
00012         _endGame = gameObject.GetComponent<EndGame>();
00013         _lifePlayersInTeamOne = GameSettingsOriginal.MaxPlayersInGame / 2;
00014         _lifePlayersInTeamTwo = GameSettingsOriginal.MaxPlayersInGame / 2;
00015     }
00016 
00017     public void PlayerInTeamOneDied()
00018     {
00019         _lifePlayersInTeamOne -= 1;
00020         if (_lifePlayersInTeamOne == 0)
00021         {
00022             _endGame.TeamOneWin();
00023         }
00024     }
00025 
00026     public void PlayerInTeamTwoDied()
00027     {
00028         _lifePlayersInTeamTwo -= 1;
00029         if (_lifePlayersInTeamTwo == 0)
00030         {
00031             _endGame.TeamTwoWin();
00032         }
00033     }
00034 
00035 
00036 }
```

7.53 Connect.cs

```
00001 using Photon.Pun;
00002 
00003 public class Connect : MonoBehaviourPunCallbacks
00004 {
00005     private Controllers _controllers;
00006 
00007     // подключение к Photon серверу
00008     void Start()
00009     {
00010         PhotonNetwork.GameVersion = "0.1";
00011         PhotonNetwork.AutomaticallySyncScene = true;
00012         PhotonNetwork.ConnectUsingSettings();
00013         _controllers = GetComponent<Controllers>();
00014 
00015     }
00016     public override void OnConnectedToMaster()
00017     {
00018         // проверка на работу fastapi сервера и авто-авторизация
00019         _controllers.CheckVujiServer();
00020     }
00021 }
```

7.54 AcceptFriendInvite.cs

```
00001 using Photon.Pun;
00002 
00003 public class AcceptFriendInvite : MonoBehaviourPunCallbacks
00004 {
00005     public string roomName;
00006 
00010     public void StartAcceptInvite()
00011     {
00012         if (PhotonNetwork.InRoom)
00013         {
00014             PhotonNetwork.LeaveRoom();
00015         }
00016         else
00017         {
00018             gameObject.GetComponent<LobbyManager>().playerStatus = "INLOBBY";
00019             PhotonNetwork.JoinRoom(roomName);
00020         }
00021     }
00022 
00023     public override void OnConnectedToMaster()
00024     {
00025         StartAcceptInvite();
```

```
00026     }
00027
00028     public override void OnJoinedRoom()
00029     {
00030         enabled = false;
00031     }
00032 }
```

7.55 FriendsListController.cs

```
00001 using UnityEngine;
00002 using UnityEngine.UI;
00003 using StructsResponse;
00004
00005 public class FriendsListController : MonoBehaviour
00006 {
00007     #region Fields
00008
00009     [SerializeField] private GameObject uiFriendsList;
00010     [SerializeField] private InputField friendsNameInputField;
00011     [SerializeField] private Transform uiFriendListScrollContent;
00012     [SerializeField] private GameObject uiFriendItemPrefab;
00013     private Controllers _controllers;
00014
00015     #endregion
00016
00017
00018     #region Unity Methods
00019
00020     private void Start()
00021     {
00022         _controllers = GetComponent<Controllers>();
00023     }
00024
00025     #endregion
00026
00027
00028     #region Public Methods
00029
00033     public void OpenFriendsList()
00034     {
00035         if (uiFriendsList.activeSelf)
00036         {
00037             uiFriendsList.SetActive(false);
00038         }
00039         else
00040         {
00041             uiFriendsList.SetActive(true);
00042         }
00043     }
00044
00049     public void FindFriendsByName()
00050     {
00051         var friendsName = friendsNameInputField.text;
00052         _controllers.FindFriendsByName(friendsName);
00053     }
00054
00059     public void FillFriendsList(UserInfoObject[] userInfoObjects)
00060     {
00061         // удалить всех старых пользователей
00062         foreach (Transform child in uiFriendListScrollContent)
00063         {
00064             Destroy(child.gameObject);
00065         }
00066
00067         // добавить новых пользователей
00068         foreach (var userinfo in userInfoObjects)
00069         {
00070             // берем префаб и заполняем его поля
00071             var friendItem = uiFriendItemPrefab.gameObject.GetComponent<FriendItemManager>();
00072             friendItem.userID = userinfo.userID;
00073             friendItem.usernameTextField.text = userinfo.username;
00074             friendItem.lobbyManager = gameObject.GetComponent<LobbyManager>();
00075
00076             // инициализируем поле установив его родителя как ScrollContent
00077             var instance = Instantiate(friendItem.gameObject);
00078             instance.transform.SetParent(uiFriendListScrollContent.transform, false);
00079         }
00080     }
00081
00082     #endregion
00083 }
```

7.56 InviteFriend.cs

```
00001 using Photon.Pun;
00002
00003 public class InviteFriend : MonoBehaviourPunCallbacks
00004 {
00005     public int invitedUserID;
00006     public string roomName;
00007     private SocketServerController _socketServerController;
00008
00013     public void StartInviteFriend()
00014     {
00015         if (roomName != null)
00016         {
00017             _socketServerController = GetComponent<SocketServerController>();
00018             _socketServerController.StartSendInviteToSocketServer(invitedUserID, roomName);
00019             enabled = false;
00020         }
00021     }
00022
00023     public override void OnJoinedRoom()
00024     {
00025         roomName = PhotonNetwork.CurrentRoom.Name;
00026         StartInviteFriend();
00027     }
00028 }
```

7.57 LeaveRoom.cs

```
00001 using Photon.Pun;
00002 using Photon.Pun.UtilityScripts;
00003 using UnityEngine;
00004
00005 public class LeaveRoom : MonoBehaviourPunCallbacks
00006 {
00007     [SerializeField] private GameObject leaveRoomButton;
00008
00012     public void LeaveFromRoom()
00013     {
00014         if (PhotonNetwork.InRoom)
00015         {
00016             PhotonNetwork.LeaveRoom();
00017         }
00018     }
00019
00020     public void ShowLeaveRoomButton()
00021     {
00022         leaveRoomButton.SetActive(true);
00023     }
00024     public void HideLeaveRoomButton()
00025     {
00026         leaveRoomButton.SetActive(false);
00027     }
00028 }
```

7.58 LobbyManager.cs

```
00001 using UnityEngine;
00002 using UnityEngine.UI;
00003 using Photon.Pun;
00004 using Photon.Realtime;
00005 using UnityEngine.SceneManagement;
00006 using Random = UnityEngine.Random;
00007
00008
00009 public class LobbyManager : MonoBehaviourPunCallbacks
00010 {
00011     #region Fields
00012
00013     [SerializeField] private Text username;
00014     [SerializeField] GameObject settings;
00015     private Controllers _controllers;
00016     // EMPTY; INLOBBY; SEARCHGAME
00017     public string playerStatus;
00018     #endregion
00019
00020     #region Unity Methods
00021
00022     void Start()
```

```
00023     {
00024         _controllers = GetComponent<Controllers>();
00025         _controllers.SetLocalUserName(username);
00026         // username.text = PhotonNetwork.NickName; // SET TO USERNAME
00027     }
00028 
00029     #endregion
00030 
00031     #region Private Methods
00032 
00036     private void QuitGame()
00037     {
00038         Application.Quit();
00039     }
00040 
00044     private void LeaveAccount()
00045     {
00046         _controllers.UserOffline();
00047         gameObject.GetComponent<SocketServerController>().CloseConnection();
00048         SceneManager.LoadScene("Login");
00049     }
00050 
00051     #endregion
00052 
00053     #region Public Methods
00054 
00060     public void CreateInviteFriend(int invitedUserID, string roomName = null)
00061     {
00062         var inviteFriend = gameObject.GetComponent<InviteFriend>();
00063         inviteFriend.enabled = true;
00064         inviteFriend.invitedUserID = invitedUserID;
00065         inviteFriend.roomName = roomName;
00066         inviteFriend.StartInviteFriend();
00067     }
00068 
00073     public void CreateLobbyAndInviteUser(int invitedUserID)
00074     {
00075         var roomName = Random.Range(1000, 10000000).ToString();
00076         RoomOptions roomOptions = new RoomOptions() {IsVisible = false, PublishUserId = true};
00077         roomOptions.MaxPlayers = 2;
00078         CreateInviteFriend(invitedUserID);
00079         gameObject.GetComponent<LobbyManager>().playerStatus = "INLOBBY";
00080         PhotonNetwork.CreateRoom(roomName, roomOptions);
00081     }
00082 
00087     public void AcceptInviteFriend(string roomName)
00088     {
00089         var acceptFriendInvite = gameObject.GetComponent<AcceptFriendInvite>();
00090         acceptFriendInvite.enabled = true;
00091         acceptFriendInvite.roomName = roomName;
00092         acceptFriendInvite.StartAcceptInvite();
00093     }
00094 
00095     public void ToggleSettings()
00096     {
00097         settings.SetActive(!settings.activeSelf);
00098     }
00099 
00100     public override void OnJoinedRoom()
00101     {
00102         PhotonNetwork.LocalPlayer.NickName = username.text;
00103         if (playerStatus == "INLOBBY")
00104         {
00105             Debug.Log("YOU JOIN IN ROOM: " + PhotonNetwork.CurrentRoom.Name);
00106             gameObject.GetComponent<LeaveRoom>().ShowLeaveRoomButton();
00107         }else if (playerStatus == "SEARCHGAME")
00108         {
00109 
00110         }
00111     }
00112 
00113     public override void OnConnectedToMaster()
00114     {
00115         Debug.Log("YOU LEFT ROOM");
00116         gameObject.GetComponent<LeaveRoom>().HideLeaveRoomButton();
00117         gameObject.GetComponent<StartGameLevel>().enabled = false;
00118         gameObject.GetComponent<PlayersFounded>().HidePlayersFounded();
00119     }
00120 
00121     #endregion
00122 }
```

7.59 NoticeListController.cs

```
00001 using UnityEngine;
00002
00003 public class NoticeListController : MonoBehaviour
00004 {
00005     [SerializeField] private GameObject uiNoticeList;
00006     [SerializeField] private Transform uiNoticeListScrollContent;
00007     [SerializeField] private GameObject uiNoticeInvitePrefab;
00008
00009     #region Public Methods
00010
00014     public void OpenNoticeList()
00015     {
00016         uiNoticeList.SetActive(!uiNoticeList.activeSelf);
00017     }
00018
00024     public void AddInviteNotice(string inviteFromUserID, string roomName)
00025     {
00026         var noticeInvite = uiNoticeInvitePrefab.gameObject.GetComponent<NoticeInviteManager>();
00027         noticeInvite.usernameTextField.text = inviteFromUserID;
00028         noticeInvite.roomName = roomName;
00029         noticeInvite.lobbyManager = gameObject.GetComponent<LobbyManager>();
00030
00031         var instance = Instantiate(noticeInvite.gameObject);
00032         instance.transform.SetParent(uiNoticeListScrollContent.transform, false);
00033     }
00034
00035     #endregion
00036 }
```

7.60 Play.cs

```
00001 using System.Collections.Generic;
00002 using Photon.Pun;
00003 using Photon.Realtime;
00004 using UnityEngine;
00005 using PhotonHashtable = ExitGames.Client.Photon.Hashtable;
00006 using Random = UnityEngine.Random;
00007
00008 public class Play : MonoBehaviourPunCallbacks
00009 {
00010     private string _masterClientIDGame;
00011     // 0 - do not start search game room
00012     // 1 - [SOLO]
00013     // 2 - [TEAM] you master client
00014     // 3 - [TEAM] you find master step 1/2
00015     // 4 - [TEAM] you find master step 2/2
00016     private int _startMode;
00017     private readonly RoomOptions _roomOptions = new RoomOptions {MaxPlayers = 4, IsOpen = true, IsVisible = true,
      PublishUserId = true};
00018     private readonly List<string> _playerInRoom = new List<string>();
00019
00023     public void StartPlayInGame()
00024     {
00025         gameObject.GetComponent<LobbyManager>().playerStatus = "SEARCHGAME";
00026         // играет один
00027         if ((PhotonNetwork.InRoom && PhotonNetwork.PlayerList.Length == 1) || (PhotonNetwork.PlayerList.Length ==
      0))
00028         {
00029             PlayInSolo();
00030         }
00031
00032         // играет не один (запускает только лидер комнаты)
00033         if (PhotonNetwork.PlayerList.Length > 1 && PhotonNetwork.IsMasterClient)
00034         {
00035             PlayInTeam();
00036         }
00037     }
00038
00042     private void PlayInSolo()
00043     {
00044         SetPlayerTeam();
00045         _startMode = 1;
00046         gameObject.GetComponent<LobbyManager>().playerStatus = "SEARCHGAME";
00047         if (PhotonNetwork.InRoom)
00048         {
00049             PhotonNetwork.LeaveRoom();
00050         }
00051         else
00052         {
00053             GoInGame();
00054         }
```

```
00055     }
00056
00061     private void PlayInTeam()
00062     {
00063         if (PhotonNetwork.IsMasterClient)
00064         {
00065             SetPlayerTeam("team");
00066             // список userID (photon) текущих в комнате
00067             foreach (var player in PhotonNetwork.PlayerListOthers)
00068             {
00069                 _playerInRoom.Add(player.UserId);
00070             }
00071
00072
00073             var view = PhotonView.Get(this);
00074             view.RPC("JoinToMasterClientRoom", RpcTarget.Others, PhotonNetwork.MasterClient.UserId);
00075
00076             _startMode = 2;
00077             gameObject.GetComponent<LobbyManager>().playerStatus = "SEARCHGAME";
00078             PhotonNetwork.LeaveRoom();
00079         }
00080     }
00081
00085     private void SetPlayerTeam(string mode = null)
00086     {
00087         if (mode == "team")
00088         {
00089             var playerTeamIs = Random.Range(1000, 100000000).ToString();
00090             PhotonHashtable playerProperties = new PhotonHashtable();
00091             playerProperties.Add("team", playerTeamIs);
00092             foreach (var player in PhotonNetwork.PlayerList)
00093             {
00094                 player.SetCustomProperties(playerProperties);
00095             }
00096         }
00097         else if(mode == null)
00098         {
00099             Player player = PhotonNetwork.LocalPlayer;
00100             PhotonHashtable playerProperties = new PhotonHashtable();
00101             playerProperties.Add("team", "None");
00102             player.SetCustomProperties(playerProperties);
00103         }
00104
00105     }
00106
00111     [PunRPC]
00112     private void JoinToMasterClientRoom(string masterClientID)
00113     {
00114         gameObject.GetComponent<LobbyManager>().playerStatus = "SEARCHGAME";
00115         _masterClientIDGame = masterClientID;
00116         FindMasterClientRoom();
00117     }
00118
00119
00125     private void FindMasterClientRoom()
00126     {
00127         if (_masterClientIDGame != null)
00128         {
00129             _startMode = 3;
00130             PhotonNetwork.LeaveRoom();
00131         }
00132     }
00133
00138     public override void OnFriendListUpdate(List<FriendInfo> friendsInfo)
00139     {
00140         foreach (var friend in friendsInfo)
00141         {
00142             if (friend.UserId == _masterClientIDGame)
00143             {
00144                 if (friend.IsInRoom)
00145                 {
00146                     _startMode = 4;
00147                     GoInGame(friend.Room);
00148                 }
00149             }
00150         }
00151     }
00152
00153     public override void OnConnectedToMaster()
00154     {
00155         if (gameObject.GetComponent<LobbyManager>().playerStatus == "SEARCHGAME")
00156         {
00157             GoInGame();
00158         }
00159
00160     }
00161
```

```
00166     private void GoInGame(string roomID = null)
00167     {
00168 
00169         if (PhotonNetwork.IsConnected)
00170         {
00171             // Play solo
00172             if (_startMode == 1)
00173             {
00174                 gameObject.GetComponent<StartGameLevel>().enabled = true;
00175                 PhotonNetwork.JoinRandomOrCreateRoom(roomOptions: _roomOptions);
00176 
00177 
00178             }
00179 
00180             // Play team master client
00181             if (_startMode == 2)
00182             {
00183                 gameObject.GetComponent<StartGameLevel>().enabled = true;
00184                 PhotonNetwork.JoinRandomOrCreateRoom(roomOptions: _roomOptions, typedLobby: TypedLobby.Default,
00185                     expectedUsers: _playerInRoom.ToArray());
00186             }
00187 
00188             // Play team NOT a master client step 1/2
00189             if (_startMode == 3)
00190             {
00191                 PhotonNetwork.FindFriends(new string[1] {_masterClientIDGame});
00192             }
00193 
00194             // Play team NOT a master client step 2/2
00195             if (_startMode == 4)
00196             {
00197                 gameObject.GetComponent<StartGameLevel>().enabled = true;
00198                 PhotonNetwork.JoinRoom(roomID);
00199             }
00200 
00201             _startMode = 0;
00202         }
00203     }
00204 }
```

7.61 PlayersFounded.cs

```
00001 using Photon.Pun;
00002 using UnityEngine;
00003 using UnityEngine.UI;
00004 using GameSettings;
00005 public class PlayersFounded : MonoBehaviour
00006 {
00007     [SerializeField] private GameObject playersFounded;
00008     [SerializeField] private Text playersFoundedText;
00009     [SerializeField] private GameObject stopSearchGameButton;
00010 
00014     public void UpdatePlayersFounded()
00015     {
00016         playersFoundedText.text = PhotonNetwork.PlayerList.Length + " / " + GameSettingsOriginal.MaxPlayersInGame
      + " founded";
00017     }
00018 
00022     public void ShowPlayersFounded()
00023     {
00024         playersFounded.SetActive(true);
00025         stopSearchGameButton.SetActive(true);
00026         Debug.Log("SHOW here" + Time.deltaTime);
00027     }
00028 
00032     public void HidePlayersFounded()
00033     {
00034         UpdatePlayersFounded();
00035         playersFounded.SetActive(false);
00036         stopSearchGameButton.SetActive(false);
00037         Debug.Log("HIDE here" + Time.deltaTime);
00038     }
00039 }
```

7.62 SocketServerController.cs

```
00001 using UnityEngine;
00002 using System.Net;
00003 using System.Net.Sockets;
00004 using System.Text;
```

```
00005 using System.Threading;
00006 using ServersInfo;
00007
00008 public class SocketServerController : MonoBehaviour
00009 {
00010     private IPEndPoint _tcpEndPoint;
00011     private Socket _tcpSocket;
00012     private Thread _thread;
00013     private DataBase _dataBase;
00014     private NoticeListController _noticeListController;
00015     private string _myToken;
00016
00017     private void Start()
00018     {
00019         _dataBase = GetComponent<DataBase>();
00020         _myToken = _dataBase.GetToken();
00021         StartConnectToSocketServer();
00022         StartHearSocketServer();
00023         _noticeListController = gameObject.GetComponent<NoticeListController>();
00024     }
00025
00026     #region Private Methods
00027
00031     private void StartConnectToSocketServer()
00032     {
00033         _tcpEndPoint = new IPEndPoint(IPAddress.Parse(SocketServerInfo.SocketServerIP),
00034             SocketServerInfo.SocketServerPort);
00035         _tcpSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
00036         ConnectToSocketServer();
00037     }
00038
00042     private void ConnectToSocketServer()
00043     {
00044         _tcpSocket.Connect(_tcpEndPoint);
00045         byte[] data = CreateMessageForSocketServer(SocketServerInfo.CommandOpenConnect, _myToken);
00046         _tcpSocket.Send(data);
00047     }
00048
00055     private byte[] CreateMessageForSocketServer(string type, params string[] data)
00056     {
00057         var message = type + ":";
00058         foreach (var info in data)
00059         {
00060             message += info + ":";
00061         }
00062
00063         message = message.Remove(message.Length - 1);
00064         var messageByte = Encoding.UTF8.GetBytes(message);
00065         return messageByte;
00066     }
00067
00073     private void SendInviteToSocketServer(int invitedUserID, string roomName)
00074     {
00075         if (_tcpSocket.Connected)
00076         {
00077             byte[] data = CreateMessageForSocketServer(SocketServerInfo.CommandInvite,
00078                 _myToken,
00079                 invitedUserID.ToString(),
00080                 roomName);
00081             _tcpSocket.Send(data);
00082         }
00083     }
00084
00088     private void StartHearSocketServer()
00089     {
00090         _thread = new Thread(HearSocketServer);
00091         _thread.Start();
00092     }
00093
00097     private void HearSocketServer()
00098     {
00099         try
00100         {
00101             while (_tcpSocket.Connected)
00102             {
00103                 var buffer = new byte[256];
00104                 var size = 0;
00105                 var answ = new StringBuilder();
00106                 do
00107                 {
00108                     size = _tcpSocket.Receive(buffer);
00109                     answ.Append(Encoding.UTF8.GetString(buffer, 0, size));
00110                 } while (_tcpSocket.Available > 0);
00111
00112                 Debug.Log("MESSAGE NEW" + answ.ToString());
00113                 HandlerMessageFromServer(answ.ToString());
00114             }
```

```
00115         }
00116         catch (SocketException ex) when (ex.ErrorCode == 10004)
00117         {
00118             // эта ошибка может возникнуть когда в главном потоке отключаешься от SocketServer
00119         }
00120     }
00121
00126     private void HandlerMessageFromServer(string data)
00127     {
00128         string[] message = data.Split(':');
00129         string command = message[0];
00130         if (command == SocketServerInfo.CommandOpenConnectServer)
00131         {
00132             Debug.Log("You connected to server");
00133         }
00134
00135         if (command == SocketServerInfo.CommandInviteServer)
00136         {
00137             Debug.Log("You invited a user");
00138         }
00139
00140         if (command == SocketServerInfo.CommandHaveInviteServer)
00141         {
00142             string inviteFromUserID = message[1];
00143             string roomName = message[2];
00144             // позволяет вызывать методы из главного потока
00145             UnityMainThreadDispatcher.Instance()
00146                 .Enqueue(() => _noticeListController.AddInviteNotice(inviteFromUserID, roomName));
00147         }
00148     }
00149
00150     #endregion
00151
00152     #region Public Methods
00153
00159     public void StartSendInviteToSocketServer(int invitedUserID, string roomName)
00160     {
00161         _thread = new Thread(() => { SendInviteToSocketServer(invitedUserID, roomName); });
00162         _thread.Start();
00163     }
00164
00168     public void CloseConnection()
00169     {
00170         _tcpSocket.Shutdown(SocketShutdown.Both);
00171         _tcpSocket.Close();
00172     }
00173
00174     #endregion
00175 }
```

7.63 StartGameLevel.cs

```
00001 using System;
00002 using System.Collections;
00003 using GameSettings;
00004 using Photon.Pun;
00005 using Photon.Realtime;
00006 using UnityEngine;
00007
00008 public class StartGameLevel : MonoBehaviourPunCallbacks
00009 {
00010     private PlayersFounded _playersFounded;
00011
00012
00013     #region Unity Methods
00014
00015     private void Start()
00016     {
00017         _playersFounded = gameObject.GetComponent<PlayersFounded>();
00018     }
00019
00020     #endregion
00021
00022     #region Photon Methods
00023
00024     public override void OnJoinedRoom()
00025     {
00026         if (gameObject.GetComponent<LobbyManager>().playerStatus == "SEARCHGAME")
00027         {
00028             _playersFounded.ShowPlayersFounded();
00029             if (CheckPreGameRoom())
00030             {
00031                 LoadGameLevel();
00032             }
```

```
00033
00034             _playersFounded.UpdatePlayersFounded();
00035         }
00036     }
00037
00038     public override void OnPlayerEnteredRoom(Player newPlayer)
00039     {
00040         if (gameObject.GetComponent<LobbyManager>().playerStatus == "SEARCHGAME")
00041         {
00042             if (CheckPreGameRoom())
00043             {
00044                 LoadGameLevel();
00045             }
00046
00047             _playersFounded.UpdatePlayersFounded();
00048         }
00049     }
00050
00051     public override void OnPlayerLeftRoom(Player newPlayer)
00052     {
00053         _playersFounded.UpdatePlayersFounded();
00054     }
00055
00056     #endregion
00057
00058
00059     #region Private Methods
00060
00065     private bool CheckPreGameRoom()
00066     {
00067         if (PhotonNetwork.PlayerList.Length == GameSettingsOriginal.MaxPlayersInGame)
00068         {
00069             return true;
00070         }
00071         Debug.Log("Not enough players");
00072         return false;
00073     }
00074
00078     private void LoadGameLevel()
00079     {
00080         PhotonNetwork.LoadLevel("Game");
00081     }
00082
00083     #endregion
00084 }
```

7.64 StopSearchGame.cs

```
00001 using System.Collections.Generic;
00002 using Photon.Pun;
00003 using Photon.Realtime;
00004 using UnityEngine;
00005
00006 public class StopSearchGame : MonoBehaviourPunCallbacks
00007 {
00008     private string _teammateID;
00009     private int _startMode;
00010     private readonly List<string> _playerInTeam = new List<string>();
00011
00012     public void CancelSearchGame()
00013     {
00014         var myTeamID = PhotonNetwork.LocalPlayer.CustomProperties["team"].ToString();
00015         if (myTeamID != "None")
00016         {
00017             foreach (var player in PhotonNetwork.PlayerListOthers)
00018             {
00019                 var playerTeamID = player.CustomProperties["team"].ToString();
00020                 if (myTeamID == playerTeamID)
00021                 {
00022                     _playerInTeam.Add(player.UserId);
00023                     var view = PhotonView.Get(this);
00024                     view.RPC("LeaveFromSearch", RpcTarget.Others, player.UserId, PhotonNetwork.LocalPlayer.UserId);
00025                     _startMode = 2;
00026                     gameObject.GetComponent<LobbyManager>().playerStatus = "INLOBBY";
00027                     PhotonNetwork.LeaveRoom();
00028                 }
00029             }
00030         }
00031         else
00032         {
00033             gameObject.GetComponent<LobbyManager>().playerStatus = "INLOBBY";
00034             PhotonNetwork.LeaveRoom();
00035         }
00036     }
```

```
00037
00038     [PunRPC]
00039     private void LeaveFromSearch(string id, string teammateID)
00040     {
00041         if (PhotonNetwork.LocalPlayer.UserId == id)
00042         {
00043             _teammateID = teammateID;
00044             if (_teammateID != null)
00045             {
00046                 _startMode = 3;
00047                 gameObject.GetComponent<LobbyManager>().playerStatus = "INLOBBY";
00048                 gameObject.GetComponent<StartGameLevel>().enabled = false;
00049                 gameObject.GetComponent<PlayersFounded>().HidePlayersFounded();
00050                 PhotonNetwork.LeaveRoom();
00051             }
00052         }
00053     }
00054
00055     public override void OnConnectedToMaster()
00056     {
00057         JoinToFriendTeam();
00058     }
00059
00060     public override void OnFriendListUpdate(List<FriendInfo> friendsInfo)
00061     {
00062         foreach (var friend in friendsInfo)
00063         {
00064             if (friend.UserId == _teammateID)
00065             {
00066                 if (friend.IsInRoom)
00067                 {
00068                     _startMode = 4;
00069                     JoinToFriendTeam(friend.Room);
00070                 }
00071             }
00072         }
00073     }
00074
00075     private void JoinToFriendTeam(string roomID = null)
00076     {
00077         if (_startMode == 2)
00078         {
00079             var roomName = Random.Range(1000, 10000000).ToString();
00080             RoomOptions roomOptions = new RoomOptions() {IsVisible = false, PublishUserId = true};
00081             roomOptions.MaxPlayers = 2;
00082             PhotonNetwork.CreateRoom(roomName, roomOptions, expectedUsers: _playerInTeam.ToArray());
00083         }
00084
00085         if (_startMode == 3)
00086         {
00087             PhotonNetwork.FindFriends(new string[1] {_teammateID});
00088         }
00089
00090         if (_startMode == 4)
00091         {
00092             PhotonNetwork.JoinRoom(roomID);
00093         }
00094
00095         _startMode = 0;
00096     }
00097 }
```

7.65 Structs.cs

```
00001 namespace StructsRequest
00002 {
00003     [System.Serializable]
00004     public class LoginStructRequest
00005     {
00006         public string login;
00007         public string password;
00008     }
00009
00010     [System.Serializable]
00011     public class RegisterStructRequest
00012     {
00013         public string login;
00014         public string password;
00015     }
00016
00017     [System.Serializable]
00018     public class UserOnlineAndOfflineStructRequest
00019     {
00020         public string data;
```

```
00021     }
00022
00023     [System.Serializable]
00024     public class FindFriendsByNameStructRequest
00025     {
00026         public string friendsName;
00027     }
00028 }
00029
00030 namespace StructsResponse
00031 {
00032     [System.Serializable]
00033     public class TokenStructResponse
00034     {
00035         public string token;
00036     }
00037
00038     [System.Serializable]
00039     public class UserIDStructResponse
00040     {
00041         public string userID;
00042     }
00043
00044     [System.Serializable]
00045     public class UserInfoStructResponse
00046     {
00047         public string login;
00048         public string username;
00049         public string created_at;
00050     }
00051
00052     [System.Serializable]
00053     public class FindFriendsByNameStructResponse
00054     {
00055         public UserInfoObject[] friends;
00056     }
00057
00058     [System.Serializable]
00059     public class UserInfoObject
00060     {
00061         public int userID;
00062         public string username;
00063     }
00064 }
00065
00066 namespace ServersInfo
00067 {
00068     [System.Serializable]
00069     public class MainServerInfo
00070     {
00071         public static string ServerDomain = "http://77.81.229.193:8000";
00072     }
00073
00074     [System.Serializable]
00075     public class SocketServerInfo
00076     {
00077         public static string SocketServerIP = "77.81.229.193";
00078         public static int SocketServerPort = 5000;
00079         public static string CommandOpenConnect = "600"; // [КЛИЕНТ] подключение к серверу
00080         public static string CommandInvite = "700"; // [КЛИЕНТ] ответ на 601S - пользователь подключен
00081
00082         public static string
00083             CommandHaveInvite = "701"; // [КЛИЕНТ], ответ на приглашение(принял/отказался) - не используется
00084
00085
00086         public static string CommandOpenConnectServer = "600S"; // [СЕРВЕР] отвечает на 600 запрос (хорошо/
      плохо)
00087         public static string CommandInviteServer = "700S"; // [СЕРВЕР] отвечает на 700 запрос (хорошо/ плохо)
00088         public static string CommandHaveInviteServer = "701S"; // [СЕРВЕР] отправляет другому игроку приглашение
00089     }
00090 }
00091
00092 namespace GameSettings
00093 {
00094     [System.Serializable]
00095     public class GameSettingsOriginal
00096     {
00097         public const int MaxPlayersInGame = 4;
00098     }
00099 }
```

7.66 DISABLESUKA.cs

```
00001 using System.Collections;
```

```
00002 using System.Collections.Generic;
00003 using UnityEngine;
00007 public class DISABLESUKA : MonoBehaviour
00008 {
00009     [SerializeField, Tooltip("Список объектов, которые необходимо отключить")] GameObject[] allObjects;
00010     // Start is called before the first frame update
00011     void Start()
00012     {
00013         foreach (var obj in allObjects)
00014         {
00015             obj.gameObject.SetActive(false);
00016         }
00017     }
00018 
00019     // Update is called once per frame
00020     void Update()
00021     {
00022 
00023     }
00024 }
```

7.67 ENABLESUKA.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00007 public class ENABLESUKA : MonoBehaviour
00008 {
00009     [SerializeField, Tooltip("Список объектов, которые необходимо включить")] GameObject[] allObjects;
00010     // Start is called before the first frame update
00011     void Start()
00012     {
00013         foreach (var obj in allObjects)
00014         {
00015             obj.gameObject.SetActive(true);
00016         }
00017     }
00018 
00019     // Update is called once per frame
00020     void Update()
00021     {
00022 
00023     }
00024 }
```

7.68 KeyHandler.cs

```
00001 using System;
00002 using System.Linq;
00003 using System.Collections;
00004 using System.Collections.Generic;
00005 using UnityEngine;
00006 using UnityEngine.SceneManagement;
00007 
00024 public class KeyHandler : MonoBehaviour
00025 {
00026 
00027     #region Fields
00028     public static KeyHandler instance; // Объект класса, необходим для обновления  и синхронизации
00029     private bool spawnPause = true; // Индикатор паузы до и после игры
00030     [SerializeField] DataBase dataBase; // База данныъ
00031     public static List<KeyCode> AllKeys; // Список всех ключей, доступных для считывания
00032     private bool binding = false; // Индикатор паузы при изменении ключа дейсвия
00033 
00034     public static Action<string, KeyCode> keyPressed; // Событие нажатия пользователем ключей действий
00035     #region KeyFields
00036     public static string[] movementKeys; // Список ключей, связанных с движением
00037     public static string[] abilityKeys; // Список ключей, связанных с способностями персонажа
00038     public static string[] uiKeys; // Список ключей, связанных с пользовательским интерфейсом
00039     private Dictionary<string, KeyCode> keybinds = new Dictionary<string, KeyCode>(); // Спиоск всех считываемых
      ключей действий
00040     #endregion
00041     #region ManagementFields
00042     private bool paused = false; // Индикатор паузы (для меню паузы)
00043     private bool uiOpened = false; // Индикатор открытия панелей UI (для меню паузы)
00044 
00045     public static KeyCode[] numbersKeyCodes = {
00046         KeyCode.Alpha1,
00047         KeyCode.Alpha2,
00048         KeyCode.Alpha3,
```

```
00049         KeyCode.Alpha4,
00050         KeyCode.Alpha5,
00051         KeyCode.Alpha6,
00052         KeyCode.Alpha7,
00053         KeyCode.Alpha8,
00054         KeyCode.Alpha9,
00055     }; // Список всех ключей цифр
00056     #endregion
00057     #endregion
00061     private void OnDestroy()
00062     {
00063         SpawnPlayers.OnSpawn -= OnSpawn;
00064         KeybindManager.Binding -= OnBinding;
00065     }
00066 
00070     void Start()
00071     {
00072         Debug.Log("KeyHandler Started");
00073         instance = this;
00074         KeybindManager.Binding += OnBinding;
00075         SpawnPlayers.OnSpawn += OnSpawn;
00076         List<KeyCode> _allKeys = new List<KeyCode> { };
00077         foreach (KeyCode keycode in Enum.GetValues(typeof(KeyCode)))
00078         {
00079             _allKeys.Add(keycode);
00080         }
00081 
00082         AllKeys = _allKeys;
00083 
00084         List<Keybind> keybindList = dataBase.GetKeybinds();
00085 
00086         List<string> movementKeysList = new List<string>();
00087         List<string> abilityKeysList = new List<string>();
00088         List<string> uiKeysList = new List<string>();
00089 
00090         foreach (Keybind keybind in keybindList)
00091         {
00092             KeyCode thisKeyCode = (KeyCode)System.Enum.Parse(typeof(KeyCode), keybind.key);
00093             keybinds[keybind.name] = thisKeyCode;
00094             switch (keybind.category)
00095             {
00096                 case "Movement":
00097                     movementKeysList.Add(keybind.name);
00098                     break;
00099                 case "Ability":
00100                     abilityKeysList.Add(keybind.name);
00101                     break;
00102                 case "UI":
00103                     uiKeysList.Add(keybind.name);
00104                     break;
00105                 default:
00106                     Debug.Log("Keybind without a category: " + keybind.name);
00107                     break;
00108             }
00109         }
00110 
00111         movementKeys = movementKeysList.ToArray();
00112         abilityKeys = abilityKeysList.ToArray();
00113         uiKeys = uiKeysList.ToArray();
00114 
00115     }
00116 
00120     void Update()
00121     {
00122         if (binding || spawnPause) return; // Считывание нажатий происходит в KeybindManager во время изменения
        привязанной клавиши. Не считывает во время паузы
00123         if (paused) // При открытии меню паузы считываются лишь нажатия esc
00124         {
00125             if (Input.GetKeyDown(KeyCode.Escape))
00126             {
00127                 keyPressed?.Invoke("EscapeMenu", KeyCode.Escape);
00128             }
00129             return;
00130         }
00131         // Read the input keys
00132         // Detect if some of keybings are pressed and Invoke keyPressed
00133         foreach (string key in keybinds.Keys)
00134         {
00135             if (Input.GetKeyDown(keybinds[key]) && !movementKeys.Contains(key))
00136             {
00137                 keyPressed?.Invoke(key, keybinds[key]);
00138             }
00139         }
00140     }
00145     void OnSpawn(GameObject player)
00146     {
00147         spawnPause = false;
```

```
00148     }
00149
00150     #region EscapeManagementFunctions
00155     public void Pause(bool pause)
00156     {
00157
00158         paused = pause;
00159     }
00164     public bool IsPaused()
00165     {
00166         return paused;
00167     }
00172     public bool IsClearAndPlaying()
00173     {
00174         return !paused && !uiOpened && !spawnPause && !binding;
00175     }
00179     public void SetUIOpened(bool opened)
00180     {
00181         uiOpened = opened;
00182     }
00187     public bool GetUIOpened()
00188     {
00189         return uiOpened;
00190     }
00191     #endregion
00195     void OnBinding(bool isBinding)
00196     {
00197         binding = isBinding;
00198     }
00203     public Dictionary<string, KeyCode> GetKeybinds()
00204     {
00205         return keybinds;
00206     }
00212     public KeyCode GetKeybind(string name)
00213     {
00214         if (!keybinds.ContainsKey(name)) return KeyCode.None;
00215         return keybinds[name];
00216     }
00223     public bool SetKeybind(string name, KeyCode key)
00224     {
00225         if (keybinds.ContainsValue(key) && keybinds[name] != key) // Исключить повторения
00226         {
00227             return false;
00228         }
00229         keybinds[name] = key;
00230         dataBase.SetKeybind(name, key);
00231         return true;
00232     }
00238     public static string NormalizeKeybind(KeyCode code)
00239     {
00240         string keyName = code.ToString();
00241         if (keyName.StartsWith("Alpha"))
00242         {
00243             return keyName[keyName.Length - 1].ToString();
00244         }
00245         else if (keyName.StartsWith("Mouse"))
00246         {
00247             switch (keyName)
00248             {
00249                 case "Mouse0":
00250                     return "LMB";
00251                 case "Mouse1":
00252                     return "RMB";
00253                 case "Mouse2":
00254                     return "MMB";
00255                 case "Mouse3":
00256                     return "SMB 1";
00257                 case "Mouse4":
00258                     return "SMB 2";
00259                 case "Mouse5":
00260                     return "SMB 3";
00261                 case "Mouse6":
00262                     return "SMB 4";
00263             }
00264         }
00265         return keyName;
00266     }
00267 }
```

7.69 EffectsListManager.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
```

```
00007 public class EffectsListManager : MonoBehaviour
00008 {
00009     [SerializeField, Tooltip("Панель для привязки эффектов")] RectTransform effectsPanel; // Панель списка эффектов
00010     [SerializeField, Tooltip("Префаб таймера эффекта")] GameObject effectPrefab; // Префаб таймера эффекта
00011     private BaseEntity targetEntity; // Сущность, для которой необходимо отслеживать накладываемые эффекты
00012 
00016     void Start()
00017     {
00018         SpawnPlayers.OnSpawn += OnSpawn;
00019     }
00020 
00024     private void OnDestroy()
00025     {
00026         SpawnPlayers.OnSpawn -= OnSpawn;
00027         if (targetEntity != null) targetEntity.OnEffectApply -= OnEffect;
00028     }
00033     void OnSpawn(GameObject player)
00034     {
00035         targetEntity = player.GetComponent<BaseEntity>();
00036         targetEntity.OnEffectApply += OnEffect;
00037     }
00043     void OnEffect(BaseEffect effect, BaseEntity entity)
00044     {
00045         if (targetEntity != entity) return;
00046         GameObject effectTimer = Instantiate(effectPrefab);
00047         effectTimer.transform.SetParent(effectsPanel, false);
00048         effectTimer.GetComponent<EffectTimerManager>().SetEffect(effect);
00049     }
00050 
00051     // Update is called once per frame
00052     void Update()
00053     {
00054 
00055     }
00056 }
```

7.70 EffectTimerManager.cs

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using UnityEngine;
00005 using UnityEngine.UI;
00009 public class EffectTimerManager : MonoBehaviour
00010 {
00011     [SerializeField, Tooltip("Изображение для спрайта целевого эффекта")] Image targetedSprite; // Изображения для отображения спрайта эффекта
00012     [SerializeField, Tooltip("Текстовое поле для отображения оставшегося времени эффекта")] Text targetedText; // Текст таймера эффекта
00013     [SerializeField, Tooltip("Модуль для установки текста подсказки при наведении")] TooltipTextUI tooltipText; // Текст подсказки при наведении
00014 
00015 
00016     private BaseEffect targetedEffect; // отслеживаемый эффект
00017 
00018     private float timerInterval = 0f; // Продолжительность эффекта
00019     private float timerValue = 0f; // Прошедшее время действия эффекта
00020     // Start is called before the first frame update
00021     void Start()
00022     {
00023         targetedText.text = "";
00024     }
00025 
00026 
00031     public void SetEffect(BaseEffect effect)
00032     {
00033         targetedSprite.sprite = effect.effectSprite;
00034         timerInterval = effect.duration;
00035         targetedEffect = effect;
00036         tooltipText.text = "\"" + effect.effectName + "\"\n" + effect.description + "\n" + "Duration: " + effect.duration + "s";
00037     }
00038 
00042     void Update()
00043     {
00044         if (targetedEffect == null) return;
00045         timerValue += Time.deltaTime;
00046         float current = timerInterval - timerValue;
00047         if (current < 0f)
00048         {
00049             current = 0f;
00050             Destroy(gameObject);
00051         }
00052         targetedText.text = current > 0f ? Convert.ToInt32(current).ToString() + "s" : "";
```

```
00053     }
00054 }
```

7.71 EndGameManager.cs

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using UnityEngine;
00005 using UnityEngine.UI;
00006 using UnityEngine.SceneManagement;
00007 using Photon.Pun;
00008 using Photon.Pun.UtilityScripts;
00012 public class EndGameManager : MonoBehaviour
00013 {
00014     [SerializeField, Tooltip("Изображение для установки спрайта победы/поражения")] Image image; // Целевое
      изображение
00015     [SerializeField, Tooltip("Текст (Победы/поражения)")] Text text; // Целевой текст
00016     [SerializeField, Tooltip("Панель окончания игры")] GameObject panel; // Целевой объект панели
00017 
00018     [SerializeField, Tooltip("Канвас UI для уничтожения")] GameObject UiCanvas; // Объект канваса для
      уничтожения
00019     [SerializeField, Tooltip("Объект GameManager для уничтожения")] GameObject GameManager; // Объект для
      уничтожения
00020 
00024     void Start()
00025     {
00026         EndGame.OnGameEnd += OnGameEnd;
00027 
00028     }
00029 
00033     private void OnDestroy()
00034     {
00035         EndGame.OnGameEnd -= OnGameEnd;
00036     }
00041     void OnGameEnd(string teamName)
00042     {
00043         panel.SetActive(true);
00044         if (PhotonNetwork.LocalPlayer.GetPhotonTeam().Name != teamName)
00045         {
00046             image.color = Color.blue;
00047             text.text = "Victory";
00048         }
00049         else
00050         {
00051             image.color = Color.red;
00052             text.text = "Defeat";
00053         }
00054         PhotonNetwork.LocalPlayer.LeaveCurrentTeam();
00055         PhotonNetwork.LeaveRoom();
00056     }
00060     public void LeaveGame()
00061     {
00062         Destroy(UiCanvas);
00063         Destroy(GameManager);
00064         SceneManager.LoadScene("Lobby");
00065     }
00066 }
```

7.72 EntityNameManager.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 using UnityEngine.UI;
00005 using Photon.Pun;
00009 public class EntityNameManager : MonoBehaviour
00010 {
00011     [Tooltip("Целевое текстовое поле")] public Text entityName; // Целевое текстовое поле
00012 
00013     private Vector3 offset; // Отклонение от позиции сущности
00014     // Start is called before the first frame update
00015     void Start()
00016     {
00017 
00018     }
00019 
00023     void Update()
00024     {
00025         Vector3 temp = Camera.main.WorldToScreenPoint(transform.parent.parent.position + offset);
```

```
00026         entityName.transform.position = new Vector3(temp.x, temp.y + 20, 0);
00027     }
00032     public void SetOffset(Vector3 newOffset)
00033     {
00034         offset = newOffset;
00035     }
00040     public Vector3 GetOffset()
00041     {
00042         return offset;
00043     }
00044
00045 }
```

7.73 EscapeMenu.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 using UnityEngine.SceneManagement;
00005 using Photon.Pun;
00009 public class EscapeMenu : MonoBehaviour
00010 {
00011     [SerializeField, Tooltip("Панель меню паузы")] private RectTransform menuPanel; // Панель меню паузы
00012     [SerializeField, Tooltip("Панель настроек")] private RectTransform settingsPanel; // Панель настроек
00013
00017     void Start()
00018     {
00019         KeyHandler.keyPressed += KeyPressed;
00020     }
00021
00025     private void OnDestroy()
00026     {
00027         KeyHandler.keyPressed -= KeyPressed;
00028     }
00034     void KeyPressed(string name, KeyCode key)
00035     {
00036         var keyHandler = KeyHandler.instance;
00037         if (name == "EscapeMenu")
00038         {
00039             if (keyHandler.GetUIOpened())
00040             {
00041                 keyHandler.SetUIOpened(false);
00042                 return;
00043             }
00044             menuPanel.gameObject.SetActive(!menuPanel.gameObject.activeSelf);
00045             keyHandler.Pause(menuPanel.gameObject.activeSelf);
00046         }
00047     }
00051     public void LeaveLobby()
00052     {
00053         PhotonNetwork.LeaveRoom();
00054         SceneManager.LoadScene("Lobby");
00055     }
00059     public void OpenSettings()
00060     {
00061         settingsPanel.gameObject.SetActive(true);
00062         menuPanel.gameObject.SetActive(false);
00063         KeyHandler.instance.SetUIOpened(true);
00064     }
00068     public void ResumeGame()
00069     {
00070         menuPanel.gameObject.SetActive(false);
00071         KeyHandler.instance.Pause(false);
00072     }
00076     public void LeaveGame()
00077     {
00078         Application.Quit();
00079     }
00080 }
```

7.74 FriendItemManager.cs

```
00001 using UnityEngine;
00002 using UnityEngine.UI;
00003 using Photon.Pun;
00004
00008 public class FriendItemManager : MonoBehaviourPunCallbacks
00009 {
00010     #region Fields
00011
```

```
00012     public int userID; // ID целевого игрока
00013     public LobbyManager lobbyManager; // Модуль лобби
00014     [SerializeField, Tooltip("Текстовое поле для имени пользователя")] public Text usernameTextField; // Целевое
      текстовое поле для имени пользователя
00015 
00016     #endregion
00017 
00018 
00019     #region Public Methods
00020 
00024     public void InviteFriend()
00025     {
00026         if (PhotonNetwork.InRoom)
00027         {
00028             lobbyManager.CreateInviteFriend(userID, PhotonNetwork.CurrentRoom.Name);
00029         }
00030         else
00031         {
00032             lobbyManager.CreateLobbyAndInviteUser(userID);
00033         }
00034     }
00035 
00036     #endregion
00037 
00038 }
```

7.75 HealthBarManager.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 using UnityEngine.UI;
00008 public class HealthBarManager : MonoBehaviour
00009 {
00010     [Tooltip("Целевой слайдер для отображения хп")] public Slider slider; // Целевой слайдер для отображения хп
00011     [SerializeField, Tooltip("Цвет для низких значений хп")] Color Low; // Цвет для низких значений хп
00012     [SerializeField, Tooltip("Цвет для высоких значений хп")] Color High; // Цвет для высоких значений хп
00013     private Vector3 offset; // Отклюнение от позиции сущности
00014 
00018     private void Start()
00019     {
00020         GetComponentInParent<Canvas>().renderMode = RenderMode.ScreenSpaceOverlay;
00021         slider.fillRect.GetComponent<RectTransform>().offsetMin = Vector2.zero;
00022         slider.fillRect.GetComponent<RectTransform>().offsetMax = Vector2.zero;
00023         slider.fillRect.parent.GetComponent<RectTransform>().offsetMax = Vector2.zero;
00024         slider.fillRect.parent.GetComponent<RectTransform>().offsetMin = Vector2.zero;
00025     }
00026 
00032     public void SetHealth(float health, float maxHp)
00033     {
00034         gameObject.SetActive(health < maxHp);
00035         slider.value = health;
00036         slider.maxValue = maxHp;
00037         slider.fillRect.GetComponentInChildren<Image>().color = Color.Lerp(Low, High, slider.normalizedValue);
00038     }
00039 
00043     void Update()
00044     {
00045         Vector3 temp = Camera.main.WorldToScreenPoint(transform.parent.parent.position + offset);
00046         slider.transform.position = new Vector3(temp.x, temp.y, 0);
00047     }
00052     public void SetOffset(Vector3 newOffset)
00053     {
00054         offset = newOffset;
00055     }
00060     public Vector3 GetOffset()
00061     {
00062         return offset;
00063     }
00064 }
```

7.76 KeybindManager.cs

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using UnityEngine;
00005 using UnityEngine.UI;
00006 
00010 public class KeybindManager : MonoBehaviour
```

```
00011 {
00012     [SerializeField, Tooltip("Текстовое поле для названия действия")] Text keybindName; // Целевое текстовое поле для названия действия
00013     [SerializeField, Tooltip("Текстовое поле для названия ключа действия")] Text keybindKeys; // Целевое текстовое поле для названия ключа дейсвтия
00014     [SerializeField, Tooltip("Ключ действия")] KeyCode key; // Ключ дейсвтвия
00015     public static Action<KeybindManager, string, KeyCode> keyChanged; // События изменения ключа действия
00016     public static Action<bool> Binding; // Событие начала изменения ключа действия
00017 
00018     private bool binding; // Индикатор изменения значения ключа пользователем
00019     // Start is called before the first frame update
00020     void Start()
00021     {
00022 
00023     }
00024 
00025     // Update is called once per frame
00026     void Update()
00027     {
00028         if (binding)
00029         {
00030             KeyCode newKey = KeyCode.None;
00031             if (Input.GetKeyDown(KeyCode.Escape))
00032             {
00033                 binding = false;
00034                 keybindKeys.text = "None";
00035                 key = KeyCode.None;
00036                 keyChanged?.Invoke(this, keybindName.text, key);
00037                 Binding?.Invoke(false);
00038                 return;
00039             }
00040             foreach (KeyCode keyCode in KeyHandler.AllKeys.ToArray())
00041             {
00042                 if (Input.GetKeyDown(keyCode))
00043                 {
00044                     newKey = keyCode;
00045                 }
00046             }
00047             if (newKey != KeyCode.None)
00048             {
00049                 key = newKey;
00050                 keybindKeys.text = KeyHandler.NormalizeKeybind(newKey);
00051                 keyChanged?.Invoke(this, keybindName.text, key);
00052                 binding = false;
00053                 Binding?.Invoke(false);
00054             }
00055         }
00056     }
00061     public string GetName()
00062     {
00063         return keybindName.text;
00064     }
00069     public KeyCode GetKey()
00070     {
00071         return key;
00072     }
00077     public void SetName(string newName)
00078     {
00079         keybindName.text = newName;
00080     }
00085     public void SetKey(KeyCode newKey)
00086     {
00087         key = newKey;
00088         keybindKeys.text = KeyHandler.NormalizeKeybind(newKey);
00089     }
00093     public void SetKeybind()
00094     {
00095         binding = true;
00096         keybindKeys.text = "_";
00097         Binding?.Invoke(true);
00098     }
00102     public void ResetKeybind()
00103     {
00104         keybindKeys.text = "None";
00105         key = KeyCode.None;
00106     }
00107 }
```

7.77 NoticeInviteManager.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 using UnityEngine.UI;
```

```
00008 public class NoticeInviteManager : MonoBehaviour
00009 {
00010     [SerializeField, Tooltip("Имя пользователя, пригласившего в группу")] public Text usernameTextField;
00011     public string roomName;
00012     public LobbyManager lobbyManager;
00013 
00014     public void AcceptInvite()
00015     {
00016         lobbyManager.AcceptInviteFriend(roomName);
00017         Destroy(gameObject);
00018     }
00019 
00020     public void CancelInvite()
00021     {
00022         Destroy(gameObject);
00023     }
00024 }
```

7.78 SettingsManager.cs

```
00001 using System;
00002 using System.Linq;
00003 using System.Collections;
00004 using System.Collections.Generic;
00005 using UnityEngine;
00006 using UnityEngine.UI;
00010 public class SettingsManager : MonoBehaviour
00011 {
00012     #region Fields
00013     public static Action<bool> keybindMovementToggled;
00014     [SerializeField, Tooltip("Префаб для настроек управления")] GameObject Keybind;
00015     [SerializeField] DataBase dataBase;
00016     #region PanelsFields
00017     [SerializeField, Tooltip("Объект панели настроек")] GameObject settingsPanel;
00018     [SerializeField, Tooltip("Объект панели настроек видео")] GameObject videoSettingsPanel;
00019     [SerializeField, Tooltip("Объект панели настроек звука")] GameObject midSettingsPanel;
00020     [SerializeField, Tooltip("Объект панели настроек управления")] GameObject keybindsPanel;
00021     [SerializeField, Tooltip("Панель списка настроек управления движением")] RectTransform movementKeys;
00022     [SerializeField, Tooltip("Панель списка настроек управления способностями")] RectTransform abilityKeys;
00023     [SerializeField, Tooltip("Панель списка настроек управления UI")] RectTransform UIKeys;
00024     [SerializeField, Tooltip("Панель блокировки настроек движения при выключенном движении по кнопкам")]
        RectTransform movementBlocker;
00025     #endregion
00026     #region VideoSettinsFields
00027     private bool fullscreen = false;
00028     private int resolutionX = 800;
00029     private int resolutionY = 600;
00030     [SerializeField, Tooltip("Текстовое поле для значения целевых ФПС")] Text maxFpsText;
00031     [SerializeField, Tooltip("Выпадающий список для разрешений экрана")] Dropdown resolutionDropdown;
00032     [SerializeField, Tooltip("Переключатель вертикальной синхронизации")] Toggle vSyncToggle;
00033     [SerializeField, Tooltip("Переключатель статуса полнного экрана для окна игры")] Toggle fullscreenToggle;
00034     [SerializeField, Tooltip("Слайдер для изменения целевых ФПС")] Slider fpsSlider;
00035     #endregion
00036 
00037     #region KeybindsFields
00038     #endregion
00039     #endregion
00040 
00041     private void OnDestroy()
00042     {
00043         KeyHandler.keyPressed -= OnKeyPressed;
00044         KeybindManager.keyChanged -= OnKeyChanged;
00045     }
00046 
00050     void Start()
00051     {
00052         var settingsList = dataBase.GetSettings();
00053         foreach (Setting setting in settingsList)
00054         {
00055             switch (setting.name)
00056             {
00057                 case "FPS":
00058                     fpsSlider.value = Convert.ToInt32(setting.value);
00059                     ChangeFps(fpsSlider);
00060                     break;
00061                 case "VSync":
00062                     vSyncToggle.isOn = Convert.ToBoolean(setting.value);
00063                     ChangeVsync(vSyncToggle);
00064                     break;
00065                 case "Resolution":
00066                     resolutionDropdown.value = Convert.ToInt32(setting.value);
00067                     ChangeResolution(resolutionDropdown);
00068                     break;
00069                 case "Fullscreen":
```

```
00070                 fullscreenToggle.isOn = Convert.ToBoolean(setting.value);
00071                 ChangeFullscreen(fullscreenToggle);
00072                 break;
00073 
00074             }
00075         }
00076 
00077 
00078         KeyHandler.keyPressed += OnKeyPressed;
00079         KeybindManager.keyChanged += OnKeyChanged;
00080         int movY = -50, ablY = -50, UIY = -50;
00081 
00082         movementKeys.GetComponent<RectTransform>().sizeDelta = new Vector2(190, KeyHandler.movementKeys.Length
    * 50 + 25);
00083         abilityKeys.GetComponent<RectTransform>().sizeDelta = new Vector2(190, KeyHandler.abilityKeys.Length * 50 +
    25);
00084         UIKeys.GetComponent<RectTransform>().sizeDelta = new Vector2(190, KeyHandler.uiKeys.Length * 50 + 25);
00085 
00086         foreach (KeyValuePair<string, KeyCode> pair in KeyHandler.instance.GetKeybinds())
00087         {
00088             if (pair.Key == "EscapeMenu") continue;
00089             GameObject keybind = Instantiate(Keybind, new Vector3(0, 100, 0), Quaternion.identity);
00090             if (KeyHandler.movementKeys.Contains(pair.Key))
00091             {
00092                 keybind.transform.SetParent(movementKeys, false);
00093                 keybind.transform.localPosition = new Vector2(0, movY);
00094                 movY -= 50;
00095             }
00096             else if (KeyHandler.abilityKeys.Contains(pair.Key))
00097             {
00098 
00099                 keybind.transform.SetParent(abilityKeys, false);
00100                 keybind.transform.localPosition = new Vector2(0, ablY);
00101                 ablY -= 50;
00102             }
00103             else if (KeyHandler.uiKeys.Contains(pair.Key))
00104             {
00105                 keybind.transform.SetParent(UIKeys, false);
00106                 keybind.transform.localPosition = new Vector2(0, UIY);
00107                 UIY -= 50;
00108             }
00109 
00110             keybind.GetComponent<KeybindManager>().SetKey(pair.Value);
00111             keybind.GetComponent<KeybindManager>().SetName(pair.Key);
00112         }
00113     }
00114 
00115     // Update is called once per frame
00116     void Update()
00117     {
00118 
00119     }
00120 
00125     public void OnKeybindMovementToggle(Toggle keybindMvmToggle)
00126     {
00127         movementBlocker.gameObject.SetActive(!keybindMvmToggle.isOn);
00128         keybindMovementToggled?.Invoke(keybindMvmToggle.isOn);
00129     }
00130 
00131     void OnKeyPressed(string name, KeyCode keys)
00132     {
00133         if (!settingsPanel) return;
00134         if (name == "EscapeMenu")
00135         {
00136             if (!settingsPanel.activeSelf) return;
00137             settingsPanel.SetActive(false);
00138             KeyHandler.instance.Pause(false);
00139         }
00140     }
00147     void OnKeyChanged(KeybindManager keybind, string name, KeyCode key)
00148     {
00149         if (!KeyHandler.instance.SetKeybind(name, key))
00150         {
00151             keybind.ResetKeybind();
00152         }
00153     }
00154     #region VideoSettingsManager
00159     public void ChangeFullscreen(Toggle fullscreenToggle)
00160     {
00161         fullscreen = fullscreenToggle.isOn;
00162         dataBase.SetSetting("Fullscreen", fullscreen.ToString());
00163         SetScreenResolution();
00164     }
00165 
00170     public void ChangeResolution(Dropdown dropdown)
00171     {
00172         dropdown.Hide();
```

```
00173         string[] resolution = dropdown.captionText.text.ToString().Split('x'); ;
00174         Debug.Log(string.Join(" ", resolution));
00175         int.TryParse(resolution[0], out resolutionX);
00176         int.TryParse(resolution[1], out resolutionY);
00177         dataBase.SetSetting("Resolution", dropdown.value.ToString());
00178         SetScreenResolution();
00179     }
00184     public void ChangeFps(Slider fpsSlider)
00185     {
00186         Application.targetFrameRate = int.Parse(fpsSlider.value.ToString());
00187         dataBase.SetSetting("FPS", fpsSlider.value.ToString());
00188         maxFpsText.text = fpsSlider.value.ToString();
00189     }
00194     public void ChangeVsync(Toggle vsyncToggle)
00195     {
00196         QualitySettings.vSyncCount = Convert.ToInt32(vsyncToggle.isOn);
00197         dataBase.SetSetting("VSync", vsyncToggle.isOn.ToString());
00198     }
00199 
00203     private void SetScreenResolution()
00204     {
00205         Screen.SetResolution(resolutionX, resolutionY, fullscreen);
00206     }
00207     #endregion
00208     #region PanelsManager
00213     public void SwitchPanels(GameObject toPanel)
00214     {
00215         videoSettingsPanel.SetActive(false);
00216         midSettingsPanel.SetActive(false);
00217         keybindsPanel.SetActive(false);
00218         toPanel.SetActive(true);
00219     }
00220     #endregion
00221 }
```

7.79 SkillPanelManager.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00007 public class SkillPanelManager : MonoBehaviour
00008 {
00009     [SerializeField, Tooltip("Префаб отображаемого скила")] GameObject timerPrefab;
00010     [SerializeField, Tooltip("Панель для привязки отображаемых скилов")] RectTransform skillPanel;
00011     public GameObject trackedPlayer;
00012     private BaseEntity trackedPlayerScript;
00013 
00014     private List<TimerWithSpritemanager> displayedSkills = new List<TimerWithSpritemanager>();
00015 
00016     // Start is called before the first frame update
00017     void Start()
00018     {
00019         SpawnPlayers.OnSpawn += OnSpawn;
00020     }
00021 
00022     private void OnDestroy()
00023     {
00024         SpawnPlayers.OnSpawn -= OnSpawn;
00025     }
00030     void OnSpawn(GameObject player)
00031     {
00032         trackedPlayer = player;
00033         trackedPlayerScript = player.GetComponent<BaseEntity>();
00034         foreach (BaseEntity.Skill skill in trackedPlayerScript.skills)
00035         {
00036             GameObject timer = Instantiate(timerPrefab);
00037             timer.transform.SetParent(skillPanel, false);
00038             timer.GetComponent<TimerWithSpritemanager>().SetEntity(trackedPlayer,
        skill.skill.GetComponent<BaseSkill>(), skill.key);
00039         }
00040     }
00041 
00042     // Update is called once per frame
00043     void Update()
00044     {
00045 
00046     }
00047 }
```

7.80 SoundSettings.cs

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using UnityEngine;
00005 using UnityEngine.UI;
00009 public class SoundSettings : MonoBehaviour
00010 {
00011     [SerializeField] DataBase dataBase;
00012     [SerializeField, Tooltip("Панель настроек звука")] RectTransform soundSettingsContent;
00013     [SerializeField, Tooltip("Префаб настройки звука")] GameObject soundSlider;
00014     public static Action<string, float> volumeChange;
00015     public static SoundSettings instance;
00016 
00017     // Sound volumes list
00018     Dictionary<string, float> volumeList = new Dictionary<string, float>();
00019 
00020     private void OnDestroy()
00021     {
00022         SoundSliderManager.onValueChange -= SoundSliderValueChange;
00023     }
00024 
00025     // Load all sounds
00026     void Start()
00027     {
00028         instance = this;
00029         SoundSliderManager.onValueChange += SoundSliderValueChange;
00030         int nowY = -75;
00031         var settings = dataBase.GetSettings();
00032         foreach (Setting setting in settings)
00033         {
00034             if (!setting.name.EndsWith("volume")) continue;
00035             GameObject slider = Instantiate(soundSlider);
00036             slider.transform.SetParent(soundSettingsContent, false);
00037             slider.transform.localPosition = new Vector2(335, nowY);
00038             var manager = slider.GetComponent<SoundSliderManager>();
00039             manager.SetName(setting.name);
00040             manager.SetValue(float.Parse(setting.value));
00041             volumeList[setting.name] = float.Parse(setting.value);
00042             nowY -= 50;
00043         }
00044     }
00050     public void SoundSliderValueChange(string name, float value)
00051     {
00052         volumeList[name] = value;
00053         dataBase.SetSetting(name, value.ToString());
00054         volumeChange?.Invoke(name, value);
00055     }
00061     public float GetVolume(string name)
00062     {
00063         return volumeList[name];
00064     }
00065 
00066 }
```

7.81 SoundSliderManager.cs

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using UnityEngine;
00005 using UnityEngine.UI;
00009 public class SoundSliderManager : MonoBehaviour
00010 {
00011     [SerializeField, Tooltip("Текстовое поле названия категории звука")] Text text;
00012     [SerializeField, Tooltip("Слайдер громкости звука целевой категории")] Slider slider;
00013 
00014     public static Action<string, float> onValueChange;
00015 
00016     private float oldValue;
00020     public void ValueChanged()
00021     {
00022         if (oldValue != slider.value)
00023         {
00024             onValueChange?.Invoke(text.text, slider.value);
00025             oldValue = slider.value;
00026         }
00027 
00028     }
00029 
00034     public void SetName(string newName)
00035     {
```

```
00036         text.text = newName;
00037     }
00042     public void SetValue(float value)
00043     {
00044         oldValue = value;
00045         slider.value = value;
00046     }
00047 }
```

7.82 TeamHpPanelManager.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00007 public class TeamHpPanelManager : MonoBehaviour
00008 {
00009     [SerializeField, Tooltip("Панель для отображения хп игроков команды")] RectTransform targetTransform;
00010     [SerializeField, Tooltip("Префаб слайдера для отображения хп игрока команды")] GameObject sliderPrefab;
00011 
00012     // Start is called before the first frame update
00013     void Start()
00014     {
00015         BaseEntity.teamSpawn += OnSpawn;
00016     }
00017     private void OnDestroy()
00018     {
00019         BaseEntity.teamSpawn -= OnSpawn;
00020     }
00021     void OnSpawn(BaseEntity player, string name)
00022     {
00023         GameObject slider = Instantiate(sliderPrefab);
00024         slider.transform.SetParent(targetTransform);
00025         slider.GetComponent<TeamHPBarUI>().SetEntity(player, name);
00026     }
00027 
00028     // Update is called once per frame
00029     void Update()
00030     {
00031 
00032     }
00033 }
```

7.83 TimerManager.cs

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using UnityEngine;
00005 using UnityEngine.UI;
00009 public class TimerManager : MonoBehaviour
00010 {
00011     public static Action<bool> timerEnd;
00012     [SerializeField, Tooltip("Текст таймера выбора класса")] Text timerText;
00013     [SerializeField, Tooltip("Слайдер для отображения прогресса таймера")] Slider timerSlider;
00014     [SerializeField, Tooltip("Заданное время для выбора класса")] float timerInterval;
00015     private float timerValue;
00016     // Start is called before the first frame update
00017     void Start()
00018     {
00019         timerValue = 0f;
00020         timerSlider.minValue = 0f;
00021         timerSlider.maxValue = timerInterval;
00022     }
00023 
00024     // Update is called once per frame
00025     void Update()
00026     {
00027         timerValue += Time.deltaTime;
00028         float current = timerInterval - timerValue;
00029         if (current < 0f)
00030         {
00031             timerEnd?.Invoke(true);
00032             current = 0f;
00033         }
00034         timerSlider.value = current;
00035         timerText.text = Convert.ToInt32(current).ToString();
00036 
00037     }
00038 }
```

7.84 TimerWithSpritemanager.cs

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using UnityEngine;
00005 using UnityEngine.UI;
00009 public class TimerWithSpritemanager : MonoBehaviour
00010 {
00011     [SerializeField, Tooltip("Целевое изображение для отображения спрайта скила")] Image targetedSprite;
00012     [SerializeField, Tooltip("Текстовое поле для отоюражения оставшегося времени перезарядки скила")] Text
      targetedText;
00013     [SerializeField, Tooltip("Обхект панели, закрывающей скрипт во время перезарядки")] GameObject coverPanel;
00014     [SerializeField, Tooltip("Обхект панели выбора скила")] GameObject selectionPanel;
00015     [SerializeField, Tooltip("Текстовое поле для отображения названия ключа действия для выбора скила")] Text
      keyText;
00016     [SerializeField, Tooltip("Модуль для изменения текста всплывающей подсказки")] TooltipTextUI tooltipText;
00017
00018     private GameObject targetPlayer;
00019     private BaseSkill targetSkill;
00020
00021     private float timerInterval = 0f;
00022     private float timerValue = 0f;
00023
00024     private string keyName;
00025
00026     public void SetTime(float time)
00027     {
00028         timerInterval = time;
00029         timerValue = 0f;
00030     }
00031
00032     private void OnDestroy()
00033     {
00034         if (targetSkill != null) targetSkill.onRelease += SetTime;
00035     }
00042     public void SetEntity(GameObject player, BaseSkill skill, string keyName)
00043     {
00044         targetPlayer = player;
00045         targetSkill = skill;
00046         this.keyName = keyName;
00047         targetedSprite.sprite = skill.GetSprite();
00048         keyText.text = KeyHandler.NormalizeKeybind(KeyHandler.instance.GetKeybind(keyName));
00049         targetedText.text = "";
00050         skill.onRelease += SetTime;
00051         tooltipText.text = "\"" + skill.GetName() + "\"\n" + skill.GetDescription() + "\n" + "Cast time: " +
      skill.GetCastTime().ToString() + "s\n" + "Energy cost: " + skill.GetCost().ToString() + "\n" + "Cooldown: " +
      skill.GetCooldownTime().ToString() + "s";
00052         player.GetComponent<BaseEntity>().OnSkillSelectionChange += SetSelection;
00053     }
00054
00060     void SetSelection(string skillKey, bool selected)
00061     {
00062         if (skillKey == keyName)
00063         {
00064             selectionPanel.SetActive(selected);
00065         }
00066     }
00067
00068     // Update is called once per frame
00069     void Update()
00070     {
00071         timerValue += Time.deltaTime;
00072         float current = timerInterval - timerValue;
00073         if (current < 0f)
00074         {
00075             current = 0f;
00076         }
00077         keyText.text = KeyHandler.NormalizeKeybind(KeyHandler.instance.GetKeybind(keyName));
00078         targetedText.text = current > 0f ? Convert.ToInt32(current).ToString() + "s" : "";
00079         coverPanel.SetActive(current > 0f);
00080     }
00081 }
```

7.85 UIHintManager.cs

```
00001 using UnityEngine;
00002 using UnityEngine.UI;
00003 using System.Collections;
00004
00005
00012 public class UIHintManager : MonoBehaviour
00013 {
```

```
00014
00015     public static string text;
00016     public static bool isUI;
00017
00018     public Color BGColor = Color.white;
00019     public Color textColor = Color.black;
00020     public enum ProjectMode { Tooltip3D = 0, Tooltip2D = 1 };
00021     public ProjectMode tooltipMode = ProjectMode.Tooltip3D;
00022     public int fontSize = 14;
00023     public int maxWidth = 250;
00024     public int border = 10;
00025     public RectTransform box;
00026     public RectTransform arrow;
00027     public Text boxText;
00028     public Camera _camera;
00029     public float speed = 10;
00030
00031     private Image[] img;
00032     private Color BGColorFade;
00033     private Color textColorFade;
00034
00035     void Awake()
00036     {
00037         img = new Image[2];
00038         img[0] = box.GetComponent<Image>();
00039         img[1] = arrow.GetComponent<Image>();
00040         box.sizeDelta = new Vector2(maxWidth, box.sizeDelta.y);
00041         BGColorFade = BGColor;
00042         BGColorFade.a = 0;
00043         textColorFade = textColor;
00044         textColorFade.a = 0;
00045         isUI = false;
00046         foreach (Image bg in img)
00047         {
00048             bg.color = BGColorFade;
00049         }
00050         boxText.color = textColorFade;
00051         boxText.alignment = TextAnchor.MiddleCenter;
00052     }
00053
00054     void LateUpdate()
00055     {
00056         bool show = false;
00057         boxText.fontSize = fontSize;
00058
00059         if (tooltipMode == ProjectMode.Tooltip3D)
00060         {
00061             RaycastHit hit;
00062             Ray ray = _camera.ScreenPointToRay(Input.mousePosition);
00063             if (Physics.Raycast(ray, out hit))
00064             {
00065                 if (hit.transform.GetComponent<TooltipText>())
00066                 {
00067                     text = hit.transform.GetComponent<TooltipText>().text;
00068                     show = true;
00069                 }
00070             }
00071         }
00072         else
00073         {
00074             RaycastHit2D hit = Physics2D.Raycast(_camera.ScreenToWorldPoint(Input.mousePosition), Vector2.zero);
00075             if (hit.transform != null)
00076             {
00077                 if (hit.transform.GetComponent<TooltipText>())
00078                 {
00079                     text = hit.transform.GetComponent<TooltipText>().text;
00080                     show = true;
00081                 }
00082             }
00083         }
00084
00085         boxText.text = text;
00086         float width = maxWidth;
00087         if (boxText.preferredWidth <= maxWidth - border) width = boxText.preferredWidth + border;
00088         box.sizeDelta = new Vector2(width, boxText.preferredHeight + 100 + border);
00089
00090         float arrowShift = width / 4;
00091
00092         if (show || isUI)
00093         {
00094             float arrowPositionY = -(arrow.sizeDelta.y / 2 - 1);
00095             float arrowPositionX = arrowShift;
00096
00097             float curY = Input.mousePosition.y + box.sizeDelta.y / 2 + arrow.sizeDelta.y;
00098             Vector3 arrowScale = new Vector3(1, 1, 1);
00099             if (curY + box.sizeDelta.y / 2 > Screen.height)
00100             {
```

```
00101             curY = Input.mousePosition.y - box.sizeDelta.y / 2 - arrow.sizeDelta.y;
00102             arrowPositionY = box.sizeDelta.y + (arrow.sizeDelta.y / 2 - 1);
00103             arrowScale = new Vector3(1, -1, 1);
00104         }
00105
00106         float curX = Input.mousePosition.x + arrowShift;
00107         if (curX + box.sizeDelta.x / 2 > Screen.width)
00108         {
00109             curX = Input.mousePosition.x - arrowShift;
00110             arrowPositionX = width - arrowShift;
00111         }
00112
00113         box.anchoredPosition = new Vector2(curX, curY);
00114
00115         arrow.anchoredPosition = new Vector2(arrowPositionX, arrowPositionY);
00116         arrow.localScale = arrowScale;
00117
00118         foreach (Image bg in img)
00119         {
00120             bg.color = Color.Lerp(bg.color, BGColor, speed * Time.deltaTime);
00121         }
00122         boxText.color = Color.Lerp(boxText.color, textColor, speed * Time.deltaTime);
00123     }
00124     else
00125     {
00126         foreach (Image bg in img)
00127         {
00128             bg.color = Color.Lerp(bg.color, BGColorFade, speed * Time.deltaTime);
00129         }
00130         boxText.color = Color.Lerp(boxText.color, textColorFade, speed * Time.deltaTime);
00131     }
00132 }
00133 }
```

7.86 EnergyBarUI.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 using UnityEngine.UI;
00008 public class EnergyBarUI : MonoBehaviour
00009 {
00010     [SerializeField, Tooltip("Целевой слайдер для отображения количества энергии")] Slider EnergyBar;
00011     [SerializeField, Tooltip("Целевое текстовое поле для отображения количества энергии")] Text EnergyBarText;
00012     private BaseEntity entity; // Целевая сущность
00013     // Start is called before the first frame update
00014     void Start()
00015     {
00016         EnergyBar.minValue = 0f;
00017         SpawnPlayers.OnSpawn += OnSpawn;
00018     }
00019     private void OnDestroy()
00020     {
00021         SpawnPlayers.OnSpawn -= OnSpawn;
00022     }
00027     void OnSpawn(GameObject player)
00028     {
00029         entity = player.GetComponent<BaseEntity>();
00030     }
00031
00032     // Update is called once per frame
00033     void Update()
00034     {
00035         if (entity == null)
00036         {
00037             EnergyBar.maxValue = 0f;
00038             EnergyBar.value = 0f;
00039             EnergyBarText.text = "";
00040             return;
00041         }
00042         if (entity.isDead)
00043         {
00044             EnergyBar.maxValue = 0f;
00045             EnergyBar.value = 0f;
00046             EnergyBarText.text = "";
00047             return;
00048         }
00049         EnergyBar.maxValue = entity.GetMaxEnergyPoints();
00050         EnergyBar.value = entity.GetEnergyPoints();
00051         EnergyBarText.text = entity.GetEnergyPoints().ToString() + "/" + entity.GetMaxEnergyPoints().ToString();
00052     }
00053     public void SetEntity(BaseEntity player)
00054     {
00055         entity = player;
```

```
00056     }
00057 }
```

7.87 HealthBarUI.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 using UnityEngine.UI;
00008 public class HealthBarUI : MonoBehaviour
00009 {
00010     [SerializeField, Tooltip("Целевой слайдер для отображения количества хп")] Slider HealthBar; // Целевой слайдер
00011     [SerializeField, Tooltip("Целевое текстовое поле для отображения количества хп")] Text HealthBarText; // Целевое
      текстовое поле
00012     private BaseEntity entity; // Целевая сущность
00013     // Start is called before the first frame update
00014     void Start()
00015     {
00016         HealthBar.minValue = 0f;
00017         SpawnPlayers.OnSpawn += OnSpawn;
00018     }
00019     private void OnDestroy()
00020     {
00021         SpawnPlayers.OnSpawn -= OnSpawn;
00022     }
00023     void OnSpawn(GameObject player)
00024     {
00025         entity = player.GetComponent<BaseEntity>();
00026     }
00027     // Update is called once per frame
00028     void Update()
00029     {
00030         if (entity == null)
00031         {
00032             HealthBar.maxValue = 0f;
00033             HealthBar.value = 0f;
00034             HealthBarText.text = "";
00035             return;
00036         }
00037         if (entity.isDead)
00038         {
00039             HealthBar.maxValue = 0f;
00040             HealthBar.value = 0f;
00041             HealthBarText.text = "";
00042             return;
00043         }
00044         HealthBar.maxValue = entity.GetMaxHealthPoints();
00045         HealthBar.value = entity.GetHealthPoints();
00046         HealthBarText.text = entity.GetHealthPoints().ToString() + "/" + entity.GetMaxHealthPoints().ToString();
00047     }
00052     public void SetEntity(BaseEntity player)
00053     {
00054         entity = player;
00055     }
00056 }
```

7.88 SkillCastTimer.cs

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using UnityEngine;
00005 using UnityEngine.UI;
00009 public class SkillCastTimer : MonoBehaviour
00010 {
00011     [SerializeField, Tooltip("Текстовое поле для отображения оставшегося времени")] Text timerText; // Целевой текст
      таймера
00012     [SerializeField, Tooltip("Слайдер для отображения оставшегося времени")] Slider timerSlider; // Целевой слайдер
      таймера
00013     private float timerInterval; // Время каста
00014     private float timerValue; // Прошедшее время каста
00015
00016     private void Start()
00017     {
00018         BaseSkill.onCast += StartTimer;
00019     }
00020
00021     private void OnDestroy()
00022     {
00023         BaseSkill.onCast -= StartTimer;
```

```
00024     }
00029     public void StartTimer(float time)
00030     {
00031         timerValue = 0f;
00032         timerInterval = time;
00033         timerSlider.minValue = 0f;
00034         timerSlider.maxValue = time;
00035     }
00036
00037     // Update is called once per frame
00038     void Update()
00039     {
00040         timerValue += Time.deltaTime;
00041         float current = timerInterval - timerValue;
00042         if (current < 0f)
00043         {
00044             current = 0f;
00045         }
00046         timerSlider.value = current;
00047         timerText.text = current > 0f ? Convert.ToInt32(current).ToString() + "s" : "";
00048
00049     }
00050 }
```

7.89 TeamHPBarUI.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 using UnityEngine.UI;
00008 public class TeamHPBarUI : MonoBehaviour
00009 {
00010     [SerializeField, Tooltip("Слайдер для отображения хп целевой сущности")] Slider HealthBar; // Целевой слайдер
00011     [SerializeField, Tooltip("Текстовое поле для отображения имени целевого пользователя")] Text HealthBarText; //
        Целевое текстовое поле для имени пользователя
00012     [SerializeField, Tooltip("Текстовое поле для индикации смерти целевого игрока")] Text DeadText; // Целевое поле
        для индикации смерти сущности игрока
00013     private BaseEntity entity; // Целевая сущность
00014     // Start is called before the first frame update
00015     void Start()
00016     {
00017         HealthBar.minValue = 0f;
00018     }
00019     // Update is called once per frame
00020     void Update()
00021     {
00022         if (entity == null)
00023         {
00024             HealthBar.maxValue = 0f;
00025             HealthBar.value = 0f;
00026             return;
00027         }
00028         if (entity.isDead)
00029         {
00030             HealthBar.maxValue = 0f;
00031             HealthBar.value = 0f;
00032             DeadText.text = "DEAD";
00033             return;
00034         }
00035         HealthBar.maxValue = entity.GetMaxHealthPoints();
00036         HealthBar.value = entity.GetHealthPoints();
00037     }
00043     public void SetEntity(BaseEntity player, string name)
00044     {
00045         entity = player;
00046         HealthBarText.text = name;
00047     }
00048 }
```

7.90 TooltipText.cs

```
00001 using UnityEngine;
00002 using System.Collections;
00006 public class TooltipText : MonoBehaviour
00007 {
00008     [Tooltip("Текст всплывающей подсказки")] public string text;
00009
00010 }
```

7.91 TooltipTextUI.cs

```
00001 using UnityEngine;
00002 using UnityEngine.EventSystems;
00003 using System.Collections;
00007 public class TooltipTextUI : MonoBehaviour, IPointerEnterHandler, IPointerExitHandler
00008 {
00009
00010     [Tooltip("Текст всплывающей подсказки")] public string text;
00011
00012     void IPointerEnterHandler.OnPointerEnter(PointerEventData e)
00013     {
00014         UIHintManager.text = text;
00015         UIHintManager.isUI = true;
00016     }
00017
00018     void IPointerExitHandler.OnPointerExit(PointerEventData e)
00019     {
00020         UIHintManager.isUI = false;
00021     }
00022
00023     void OnDestroy()
00024     {
00025         UIHintManager.isUI = false;
00026     }
00027 }
```

7.92 UnityMainThreadDispatcher.cs

```
00001 /*
00002 Copyright 2015 Pim de Witte All Rights Reserved.
00003
00004 Licensed under the Apache License, Version 2.0 (the "License");
00005 you may not use this file except in compliance with the License.
00006 You may obtain a copy of the License at
00007
00008     http://www.apache.org/licenses/LICENSE-2.0
00009
00010 Unless required by applicable law or agreed to in writing, software
00011 distributed under the License is distributed on an "AS IS" BASIS,
00012 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
00013 See the License for the specific language governing permissions and
00014 limitations under the License.
00015 */
00016
00017 using UnityEngine;
00018 using System.Collections;
00019 using System.Collections.Generic;
00020 using System;
00021 using System.Threading.Tasks;
00022
00028 public class UnityMainThreadDispatcher : MonoBehaviour {
00029
00030     private static readonly Queue<Action> _executionQueue = new Queue<Action>();
00031
00032     public void Update() {
00033         lock(_executionQueue) {
00034             while (_executionQueue.Count > 0) {
00035                 _executionQueue.Dequeue().Invoke();
00036             }
00037         }
00038     }
00039
00044     public void Enqueue(IEnumerator action) {
00045         lock (_executionQueue) {
00046             _executionQueue.Enqueue (() => {
00047                 StartCoroutine (action);
00048             });
00049         }
00050     }
00051
00056     public void Enqueue(Action action)
00057     {
00058         Enqueue(ActionWrapper(action));
00059     }
00060
00066     public Task EnqueueAsync(Action action)
00067     {
00068         var tcs = new TaskCompletionSource<bool>();
00069
00070         void WrappedAction() {
00071             try
00072             {
```

```
00073                 action();
00074                 tcs.TrySetResult(true);
00075             } catch (Exception ex)
00076             {
00077                 tcs.TrySetException(ex);
00078             }
00079         }
00080 
00081         Enqueue(ActionWrapper(WrappedAction));
00082         return tcs.Task;
00083     }
00084 
00085 
00086     IEnumerator ActionWrapper(Action a)
00087     {
00088         a();
00089         yield return null;
00090     }
00091 
00092 
00093     private static UnityMainThreadDispatcher _instance = null;
00094 
00095     public static bool Exists() {
00096         return _instance != null;
00097     }
00098 
00099     public static UnityMainThreadDispatcher Instance() {
00100         if (!Exists ()) {
00101             throw new Exception ("UnityMainThreadDispatcher could not find the UnityMainThreadDispatcher object.
      Please ensure you have added the MainThreadExecutor Prefab to your scene.");
00102         }
00103         return _instance;
00104     }
00105 
00106 
00107     void Awake() {
00108         if (_instance == null) {
00109             _instance = this;
00110             DontDestroyOnLoad(this.gameObject);
00111         }
00112     }
00113 
00114     void OnDestroy() {
00115             _instance = null;
00116     }
00117 
00118 
00119 }
```

7.93 UserOnline.cs

```
00001 using UnityEngine;
00005 public class UserOnline : MonoBehaviour
00006 {
00007     private Controllers _controllers;
00008     private float _userTimeOnline = 5;
00009     private float _currentUserTimeOnline;
00010 
00011     #region Unity Methods
00012 
00013     void Start()
00014     {
00015         _controllers = gameObject.AddComponent<Controllers>();
00016         _currentUserTimeOnline = _userTimeOnline;
00017     }
00018 
00019     #endregion
00020 
00021     #region Private Methods
00022 
00026     private void Update()
00027     {
00028         _currentUserTimeOnline -= Time.deltaTime;
00029         if (_currentUserTimeOnline <= 0)
00030         {
00031             _controllers.UserOnline();
00032             _currentUserTimeOnline = _userTimeOnline;
00033         }
00034     }
00035 
00036     #endregion
00037 }
```

Предметный указатель

_attack
AnimationPlayer, 19
_camera
UIHintManager, 162
_controller
BaseEntity, 33
_down
AnimationPlayer, 19
_drink
AnimationPlayer, 19
_droppedItemPrefab
BaseEntity, 33
_idle
AnimationPlayer, 19
_left
AnimationPlayer, 19
_move
AnimationPlayer, 19
_right
AnimationPlayer, 19
_shot
AnimationPlayer, 20
_up
AnimationPlayer, 20

abilityKeys
KeyHandler, 102
AcceptFriendInvite, 15
OnConnectedToMaster, 15
OnJoinedRoom, 16
roomName, 16
StartAcceptInvite, 16
AcceptInvite
NoticeInviteManager, 113
AcceptInviteFriend
LobbyManager, 105
ActivatePassiveSkill
BasePassiveSkill, 37
AddDamage
BaseProjectile, 39
AddEffect
BaseEntity, 27
AddInviteNotice
NoticeListController, 114
AddItem
Inventory, 89
additionalDamage
Strong, 154
additionalDefense
Drunk, 63
additionalSpeed
SpeedUp, 149
AddKeybind
DataBase, 53
AddSetting
DataBase, 54
AddUpgrade
PlayerUpgrades, 125
AddXp
PlayerUpgrades, 125
AgressionStart
EnemyAI, 70
AgressionTrigger, 17
AllKeys
KeyHandler, 102
AnimationPlayer, 17
_attack, 19
_down, 19
_drink, 19
_idle, 19
_left, 19
_move, 19
_right, 19
_shot, 20
_up, 20
ChangePlayerAnimation, 18
ChangePlayerAnimation_q, 18
getCurrentMovingState, 18
movingState, 20
AOE
PlayerAOE.AOEstruct, 21
AOEKey
PlayerAOE.AOEstruct, 21
Appe, 21
UseItem, 21
ApplyEffect
BaseEffect, 25
Drunk, 63
OnFire, 116
Poison, 127
SpeedUp, 149
Strong, 154
ApplyUpgrade
BaseUpgrade, 45
arrow
UIHintManager, 163
Attack
EntityMelee, 71

PlayerAOE, 119
PlayerProjectile, 120, 121
AttackTrigger, 22

BaseAOE, 22
GetAoeDescription, 23
GetAoeName, 23
SetDamageTags, 23
SetDestroyTags, 23
SetSenderCollider, 24
BaseEffect, 24
ApplyEffect, 25
description, 25
duration, 25
effectName, 25
effectSprite, 25
BaseEntity, 26
_controller, 33
_droppedItemPrefab, 33
AddEffect, 27
DecreaseDamage, 27
DecreaseDefense, 27
DecreaseSpeed, 28
deSelectSkill, 28
displayedName, 33
GetBaseDamage, 28
GetDefense, 28
GetEnergyPoints, 28
GetEntityName, 29
GetHealthPoints, 29
GetMaxEnergyPoints, 29
GetMaxHealthPoints, 29
GetMoveSpeed, 29
GetSelectedSkill, 30
Heal, 30
healthBar, 34
IncreaseDamage, 30
IncreaseDefense, 30
IncreaseMaxEnergy, 30
IncreaseMaxHealth, 31
IncreaseSpeed, 31
isDead, 34
OnEffectApply, 34
OnSkillSelectionChange, 34
selectSkill, 31
setIsCooldown, 31
skills, 34
spendEnergy, 32
TakeDamage, 32
teamSpawn, 34
TickPoints, 32
UpdateText, 32
UseSkill, 33
BaseEntity.Skill, 137
key, 137
skill, 137
BaseItem, 35
GetAmount, 35
GetDescription, 35
GetImage, 36
GetItemName, 36
SetAmount, 36
UseItem, 36
BasePassiveSkill, 37
ActivatePassiveSkill, 37
GetDescription, 38
GetName, 38
skillDescription, 38
skillName, 38
BaseProjectile, 39
AddDamage, 39
GetProjectileDescription, 39
GetProjectileName, 39
SetAimAngel, 40
SetAimDirection, 40
SetDamageTags, 40
SetDestroyTags, 40
SetSenderCollider, 40
BaseSkill, 41
cancelMovementOnCast, 43
castTime, 43
cooldown, 43
energyCost, 44
GetCastTime, 42
GetCooldownTime, 42
GetCost, 42
GetDescription, 42
GetName, 42
GetSprite, 43
onCast, 44
onRelease, 44
skillDescription, 44
skillName, 44
skillSprite, 44
UseSkill, 43
BaseUpgrade, 45
ApplyUpgrade, 45
GetCost, 45
GetDescription, 46
GetName, 46
Berserk, 46
BGColor
UIHintManager, 163
Binding
KeybindManager, 97
border
UIHintManager, 163
box
UIHintManager, 163
boxText
UIHintManager, 163

CameraFollow, 47
player, 47
CameraPlayer, 47
CancelInvite
NoticeInviteManager, 113
cancelMovement

MovementPlayer, 112
cancelMovementOnCast
BaseSkill, 43
CancelSearchGame
StopSearchGame, 152
canMove
MovementPlayer, 112
castTime
BaseSkill, 43
category
Keybind, 94
ChangeFps
SettingsManager, 134
ChangeFullscreen
SettingsManager, 134
ChangePlayerAnimation
AnimationPlayer, 18
ChangePlayerAnimation_q
AnimationPlayer, 18
ChangeResolution
SettingsManager, 135
ChangeVsync
SettingsManager, 135
CheckVujiServer
Controllers, 50
ClassSelection, 48
ClearInventory
Inventory, 89
CloseConnection
SocketServerController, 140
CommandHaveInvite
ServersInfo.SocketServerInfo, 141
CommandHaveInviteServer
ServersInfo.SocketServerInfo, 141
CommandInvite
ServersInfo.SocketServerInfo, 141
CommandInviteServer
ServersInfo.SocketServerInfo, 141
CommandOpenConnect
ServersInfo.SocketServerInfo, 141
CommandOpenConnectServer
ServersInfo.SocketServerInfo, 141
Connect, 48
OnConnectedToMaster, 49
Controllers, 49
CheckVujiServer, 50
FindFriendsByName, 50
GetUserID, 50
Login, 51
Register, 51
SetLocalUserName, 51
UserOffline, 52
UserOnline, 52
cooldown
BaseSkill, 43
created_at
StructsResponse.UserInfoStructResponse, 169
CreateInviteFriend
LobbyManager, 106
CreateLobbyAndInviteUser
LobbyManager, 106

damageTickSeconds
Poison, 127
data
StructsRequest.UserOnlineAndOfflineStructRequest, 171
DataBase, 53
AddKeybind, 53
AddSetting, 54
ExistKeybind, 54
ExistSetting, 55
GetKeybinds, 55
GetSettings, 56
GetToken, 56
SetKeybind, 57
SetSetting, 57
SetToken, 58
DecreaseDamage
BaseEntity, 27
DecreaseDefense
BaseEntity, 27
DecreaseSpeed
BaseEntity, 28
description
BaseEffect, 25
deSelectSkill
BaseEntity, 28
DISABLESUKA, 58
DisplayedItem, 59
displayedItem, 60
inventoryPanel, 60
itemId, 60
itemPosition, 61
onItemDrop, 61
onItemSwap, 61
OnPointerDown, 59
OnPointerUp, 60
displayedItem
DisplayedItem, 60
displayedName
BaseEntity, 33
DropItem
Inventory, 89
DroppedItem, 61
itemData, 62
SetItem, 62
Drunk, 62
additionalDefense, 63
ApplyEffect, 63
DrunkEffect, 63
Drunkard, 64
UseSkill, 64
DrunkEffect
Drunk, 63
duration
BaseEffect, 25

effectName
BaseEffect, 25
EffectsListManager, 65
effectSprite
BaseEffect, 25
EffectTimerManager, 66
SetEffect, 66
ENABLESUKA, 67
EndGame, 67
OnGameEnd, 68
TeamOneWin, 68
TeamTwoWin, 68
EndGameManager, 68
LeaveGame, 69
EnemyAI, 69
AgressionStart, 70
reachedEndOfPath, 70
EnergyBarUI, 70
SetEntity, 71
energyCost
BaseSkill, 44
Enqueue
UnityMainThreadDispatcher, 166
EnqueueAsync
UnityMainThreadDispatcher, 166
EntityMelee, 71
Attack, 71
entityName
EntityNameManager, 73
EntityNameManager, 72
entityName, 73
GetOffset, 73
SetOffset, 73
EscapeMenu, 74
LeaveGame, 74
LeaveLobby, 74
OpenSettings, 75
ResumeGame, 75
Exclamation, 75
UseSkill, 76
ExistKeybind
DataBase, 54
Exists
UnityMainThreadDispatcher, 167
ExistSetting
DataBase, 55

FillFriendsList
FriendsListController, 82
FindFriendsByName
Controllers, 50
FriendsListController, 82
FireballSkill, 78
UseSkill, 78
FlurryOfFire, 79
UseSkill, 79
fontSize
UIHintManager, 163
FriendItemManager, 80
InviteFriend, 80
lobbyManager, 81
userID, 81
usernameTextField, 81
friends
StructsResponse.FindFriendsByNameStructResponse, 77
FriendsListController, 81
FillFriendsList, 82
FindFriendsByName, 82
OpenFriendsList, 82
friendsName
StructsRequest.FindFriendsByNameStructRequest, 77

GameSettings, 13
GameSettings.GameSettingsOriginal, 83
MaxPlayersInGame, 83
Generator, 84
Height, 84
RoomPrefabs, 84
StartingRoom, 84
Width, 84
GetAllItems
Inventory, 89
GetAmount
BaseItem, 35
GetAoeDescription
BaseAOE, 23
GetAoeName
BaseAOE, 23
GetBaseDamage
BaseEntity, 28
GetCastTime
BaseSkill, 42
GetCooldownTime
BaseSkill, 42
GetCost
BaseSkill, 42
BaseUpgrade, 45
getCurrentMovingState
AnimationPlayer, 18
GetDefense
BaseEntity, 28
GetDescription
BaseItem, 35
BasePassiveSkill, 38
BaseSkill, 42
BaseUpgrade, 46
GetEnergyPoints
BaseEntity, 28
GetEntityName
BaseEntity, 29
GetHealthPoints
BaseEntity, 29
GetImage
BaseItem, 36
GetItemName
BaseItem, 36

GetKey
KeybindManager, 95
GetKeybind
KeyHandler, 99
GetKeybinds
DataBase, 55
KeyHandler, 99
GetMaxEnergyPoints
BaseEntity, 29
GetMaxHealthPoints
BaseEntity, 29
GetMoveSpeed
BaseEntity, 29
GetName
BasePassiveSkill, 38
BaseSkill, 42
BaseUpgrade, 46
KeybindManager, 95
GetOffset
EntityNameManager, 73
HealthBarManager, 85
GetProjectileDescription
BaseProjectile, 39
GetProjectileName
BaseProjectile, 39
GetSelectedSkill
BaseEntity, 30
GetSettings
DataBase, 56
GetSprite
BaseSkill, 43
GetToken
DataBase, 56
GetUIOpened
KeyHandler, 100
GetUserID
Controllers, 50
GetVolume
SoundSettings, 143
GetXp
PlayerUpgrades, 126
goblinSeekerGameObject2
SpawnEnemy, 147

Heal
BaseEntity, 30
healthBar
BaseEntity, 34
HealthBarManager, 85
GetOffset, 85
SetHealth, 86
SetOffset, 86
slider, 87
HealthBarUI, 87
SetEntity, 87
Height
Generator, 84
HideLeaveRoomButton
LeaveRoom, 104
HidePlayersFounded
PlayersFounded, 123

IncreaseDamage
BaseEntity, 30
IncreaseDefense
BaseEntity, 30
IncreaseMaxEnergy
BaseEntity, 30
IncreaseMaxHealth
BaseEntity, 31
IncreaseSpeed
BaseEntity, 31
InputControllerPlayer, 88
Instance
UnityMainThreadDispatcher, 167
instance
KeyHandler, 102
SoundSettings, 144
Inventory, 88
AddItem, 89
ClearInventory, 89
DropItem, 89
GetAllItems, 89
inventoryItems, 90
onItemAdded, 90
SyncronisedDrop, 90
inventoryItems
Inventory, 90
inventoryPanel
DisplayedItem, 60
InventoryWindow, 91
OnSpawn, 91
Start, 91
invitedUserID
InviteFriend, 93
InviteFriend, 92
FriendItemManager, 80
invitedUserID, 93
OnJoinedRoom, 92
roomName, 93
StartInviteFriend, 92
IsClearAndPlaying
KeyHandler, 100
isDead
BaseEntity, 34
IsPaused
KeyHandler, 100
isUI
UIHintManager, 164
itemData
DroppedItem, 62
itemId
DisplayedItem, 60
itemPosition
DisplayedItem, 61

key
BaseEntity.Skill, 137

Keybind, 94
Keybind, 93
category, 94
key, 94
Keybind, 93
name, 94
KeybindManager, 94
Binding, 97
GetKey, 95
GetName, 95
keyChanged, 97
ResetKeybind, 96
SetKey, 96
SetKeybind, 96
SetName, 97
keybindMovementToggled
SettingsManager, 136
keyChanged
KeybindManager, 97
KeyHandler, 98
abilityKeys, 102
AllKeys, 102
GetKeybind, 99
GetKeybinds, 99
GetUIOpened, 100
instance, 102
IsClearAndPlaying, 100
IsPaused, 100
keyPressed, 103
movementKeys, 103
NormalizeKeybind, 100
numbersKeyCodes, 103
Pause, 101
SetKeybind, 101
SetUIOpened, 102
uiKeys, 103
keyPressed
KeyHandler, 103
KillPlayer
PlayerScript, 122

LeaveFromRoom
LeaveRoom, 104
LeaveGame
EndGameManager, 69
EscapeMenu, 74
LeaveLobby
EscapeMenu, 74
LeaveRoom, 104
HideLeaveRoomButton, 104
LeaveFromRoom, 104
ShowLeaveRoomButton, 104
LobbyManager, 105
AcceptInviteFriend, 105
CreateInviteFriend, 106
CreateLobbyAndInviteUser, 106
OnConnectedToMaster, 107
OnJoinedRoom, 107
playerStatus, 107
ToggleSettings, 107
lobbyManager
FriendItemManager, 81
NoticeInviteManager, 113
Login
Controllers, 51
login
StructsRequest.LoginStructRequest, 109
StructsRequest.RegisterStructRequest, 131
StructsResponse.UserInfoStructResponse, 170
LoginInAccount
LoginManager, 108
loginInput
LoginManager, 109
RegisterManager, 130
LoginManager, 108
LoginInAccount, 108
loginInput, 109
passwordInput, 109
ShowRegisterScene, 108

ManagerGame, 110
OnPlayerPropertiesUpdate, 111
MasterCheckMeleeAttack
PlayerMelee, 119
MaxPlayersInGame
GameSettings.GameSettingsOriginal, 83
maxWidth
UIHintManager, 164
movementKeys
KeyHandler, 103
MovementPlayer, 111
cancelMovement, 112
canMove, 112
movingState
AnimationPlayer, 20

name
Keybind, 94
Setting, 133
NormalizeKeybind
KeyHandler, 100
NoticeInviteManager, 112
AcceptInvite, 113
CancelInvite, 113
lobbyManager, 113
roomName, 114
usernameTextField, 114
NoticeListController, 114
AddInviteNotice, 114
OpenNoticeList, 115
numbersKeyCodes
KeyHandler, 103

onCast
BaseSkill, 44
OnConnectedToMaster
AcceptFriendInvite, 15
Connect, 49

LobbyManager, 107
Play, 117
StopSearchGame, 152
OnEffectApply
BaseEntity, 34
OnFire, 115
ApplyEffect, 116
OnFireEffect, 116
OnFireEffect
OnFire, 116
OnFriendListUpdate
Play, 117
StopSearchGame, 153
OnGameEnd
EndGame, 68
onItemAdded
Inventory, 90
onItemDrop
DisplayedItem, 61
onItemSwap
DisplayedItem, 61
OnJoinedRoom
AcceptFriendInvite, 16
InviteFriend, 92
LobbyManager, 107
StartGameLevel, 151
OnKeybindMovementToggle
SettingsManager, 136
OnPlayerEnteredRoom
StartGameLevel, 151
OnPlayerLeftRoom
StartGameLevel, 151
OnPlayerPropertiesUpdate
ManagerGame, 111
OnPointerDown
DisplayedItem, 59
OnPointerUp
DisplayedItem, 60
onRelease
BaseSkill, 44
OnSkillSelectionChange
BaseEntity, 34
OnSpawn
InventoryWindow, 91
SpawnPlayers, 148
onValueChange
SoundSliderManager, 146
OpenFriendsList
FriendsListController, 82
OpenNoticeList
NoticeListController, 115
OpenSettings
EscapeMenu, 75

password
StructsRequest.LoginStructRequest, 109
StructsRequest.RegisterStructRequest, 131
passwordInput
LoginManager, 109
passwordOneInput
RegisterManager, 130
passwordTwoInput
RegisterManager, 130
Pause
KeyHandler, 101
Play, 116
OnConnectedToMaster, 117
OnFriendListUpdate, 117
StartPlayInGame, 118
player
CameraFollow, 47
PlayerAOE, 118
Attack, 119
PlayerAOE.AOEstruct, 20
AOE, 21
AOEKey, 21
PlayerInTeamOneDied
PlayersTeamsManager, 124
PlayerInTeamTwoDied
PlayersTeamsManager, 124
PlayerMelee, 119
MasterCheckMeleeAttack, 119
PlayerProjectile, 120
Attack, 120, 121
PlayerProjectile.Projectile, 128
projectile, 128
projectileKey, 128
PlayerScript, 121
KillPlayer, 122
PlayersFounded, 122
HidePlayersFounded, 123
ShowPlayersFounded, 123
UpdatePlayersFounded, 123
playerStatus
LobbyManager, 107
PlayersTeamsManager, 124
PlayerInTeamOneDied, 124
PlayerInTeamTwoDied, 124
PlayerUpgrades, 125
AddUpgrade, 125
AddXp, 125
GetXp, 126
SwitchPanels, 126
Poison, 126
ApplyEffect, 127
damageTickSeconds, 127
PoisonEffect, 127
posionDamage, 128
PoisonEffect
Poison, 127
posionDamage
Poison, 128
projectile
PlayerProjectile.Projectile, 128
projectileKey
PlayerProjectile.Projectile, 128
ProjectMode

UIHintManager, 162

reachedEndOfPath
EnemyAI, 70
Register
Controllers, 51
RegisterAccount
RegisterManager, 129
RegisterManager, 129
loginInput, 130
passwordOneInput, 130
passwordTwoInput, 130
RegisterAccount, 129
ShowLoginScene, 129
ResetKeybind
KeybindManager, 96
ResumeGame
EscapeMenu, 75
Room, 131
RotateRandomly, 132
roomName
AcceptFriendInvite, 16
InviteFriend, 93
NoticeInviteManager, 114
RoomPrefabs
Generator, 84
RotateRandomly
Room, 132

selectSkill
BaseEntity, 31
ServerDomain
ServersInfo.MainServerInfo, 110
ServersInfo, 13
ServersInfo.MainServerInfo, 110
ServerDomain, 110
ServersInfo.SocketServerInfo, 140
CommandHaveInvite, 141
CommandHaveInviteServer, 141
CommandInvite, 141
CommandInviteServer, 141
CommandOpenConnect, 141
CommandOpenConnectServer, 141
SocketServerIP, 142
SocketServerPort, 142
SetAimAngel
BaseProjectile, 40
SetAimDirection
BaseProjectile, 40
SetAmount
BaseItem, 36
SetDamageTags
BaseAOE, 23
BaseProjectile, 40
SetDestroyTags
BaseAOE, 23
BaseProjectile, 40
SetEffect
EffectTimerManager, 66
SetEntity
EnergyBarUI, 71
HealthBarUI, 87
TeamHPBarUI, 155
TimerWithSpritemanager, 157
SetHealth
HealthBarManager, 86
setIsCooldown
BaseEntity, 31
SetItem
DroppedItem, 62
SetKey
KeybindManager, 96
SetKeybind
DataBase, 57
KeybindManager, 96
KeyHandler, 101
SetLocalUserName
Controllers, 51
SetName
KeybindManager, 97
SoundSliderManager, 145
SetOffset
EntityNameManager, 73
HealthBarManager, 86
SetPlayerObject
SpawnPlayers, 148
SetSenderCollider
BaseAOE, 24
BaseProjectile, 40
SetSetting
DataBase, 57
SetTime
TimerWithSpritemanager, 158
Setting, 132
name, 133
Setting, 132
value, 133
SettingsManager, 133
ChangeFps, 134
ChangeFullscreen, 134
ChangeResolution, 135
ChangeVsync, 135
keybindMovementToggled, 136
OnKeybindMovementToggle, 136
SwitchPanels, 136
SetToken
DataBase, 58
SetUIOpened
KeyHandler, 102
SetValue
SoundSliderManager, 146
ShowLeaveRoomButton
LeaveRoom, 104
ShowLoginScene
RegisterManager, 129
ShowPlayersFounded
PlayersFounded, 123

ShowRegisterScene
LoginManager, 108
skill
BaseEntity.Skill, 137
SkillCastTimer, 137
StartTimer, 138
skillDescription
BasePassiveSkill, 38
BaseSkill, 44
skillName
BasePassiveSkill, 38
BaseSkill, 44
SkillPanelManager, 138
trackedPlayer, 139
skills
BaseEntity, 34
skillSprite
BaseSkill, 44
slider
HealthBarManager, 87
SocketServerController, 139
CloseConnection, 140
StartSendInviteToSocketServer, 140
SocketServerIP
ServersInfo.SocketServerInfo, 142
SocketServerPort
ServersInfo.SocketServerInfo, 142
SoundManager, 142
SoundSettings, 143
GetVolume, 143
instance, 144
SoundSliderValueChange, 144
volumeChange, 144
SoundSliderManager, 145
onValueChange, 146
SetName, 145
SetValue, 146
ValueChanged, 146
SoundSliderValueChange
SoundSettings, 144
SpawnEnemy, 147
goblinSeekerGameObject2, 147
SpawnPlayers, 147
OnSpawn, 148
SetPlayerObject, 148
speed
UIHintManager, 164
SpeedEffect
SpeedUp, 149
SpeedUp, 148
additionalSpeed, 149
ApplyEffect, 149
SpeedEffect, 149
spendEnergy
BaseEntity, 32
Start
InventoryWindow, 91
StartAcceptInvite
AcceptFriendInvite, 16
StartEndZone, 150
StartGameLevel, 150
OnJoinedRoom, 151
OnPlayerEnteredRoom, 151
OnPlayerLeftRoom, 151
StartingRoom
Generator, 84
StartInviteFriend
InviteFriend, 92
StartPlayInGame
Play, 118
StartSendInviteToSocketServer
SocketServerController, 140
StartTimer
SkillCastTimer, 138
StopSearchGame, 152
CancelSearchGame, 152
OnConnectedToMaster, 152
OnFriendListUpdate, 153
Strong, 153
additionalDamage, 154
ApplyEffect, 154
StrongEffect, 154
StrongEffect
Strong, 154
StructsRequest, 13
StructsRequest.FindFriendsByNameStructRequest, 76
friendsName, 77
StructsRequest.LoginStructRequest, 109
login, 109
password, 109
StructsRequest.RegisterStructRequest, 130
login, 131
password, 131
StructsRequest.UserOnlineAndOfflineStructRequest, 171
data, 171
StructsResponse, 13
StructsResponse.FindFriendsByNameStructResponse, 77
friends, 77
StructsResponse.TokenStructResponse, 158
token, 158
StructsResponse.UserIDStructResponse, 168
userID, 168
StructsResponse.UserInfoObject, 168
userID, 169
username, 169
StructsResponse.UserInfoStructResponse, 169
created_at, 169
login, 170
username, 170
SwitchPanels
PlayerUpgrades, 126
SettingsManager, 136
SyncronisedDrop

Inventory, 90

TakeDamage
BaseEntity, 32
TeamHPBarUI, 155
SetEntity, 155
TeamHpPanelManager, 156
TeamOneWin
EndGame, 68
teamSpawn
BaseEntity, 34
TeamTwoWin
EndGame, 68
text
TooltipText, 159
TooltipTextUI, 160
UIHintManager, 164
textColor
UIHintManager, 164
TickPoints
BaseEntity, 32
timerEnd
TimerManager, 156
TimerManager, 156
timerEnd, 156
TimerWithSpritemanager, 157
SetEntity, 157
SetTime, 158
ToggleSettings
LobbyManager, 107
token
StructsResponse.TokenStructResponse, 158
tooltipMode
UIHintManager, 164
TooltipText, 159
text, 159
TooltipTextUI, 160
text, 160
Toxicant, 160
UseSkill, 161
trackedPlayer
SkillPanelManager, 139

UIHintManager, 161
_camera, 162
arrow, 163
BGColor, 163
border, 163
box, 163
boxText, 163
fontSize, 163
isUI, 164
maxWidth, 164
ProjectMode, 162
speed, 164
text, 164
textColor, 164
tooltipMode, 164
uiKeys
KeyHandler, 103
UnityMainThreadDispatcher, 165
Enqueue, 166
EnqueueAsync, 166
Exists, 167
Instance, 167
Update, 167
Update
UnityMainThreadDispatcher, 167
UpdatePlayersFounded
PlayersFounded, 123
UpdateText
BaseEntity, 32
UseItem
Appe, 21
BaseItem, 36
userID
FriendItemManager, 81
StructsResponse.UserIDStructResponse, 168
StructsResponse.UserInfoObject, 169
username
StructsResponse.UserInfoObject, 169
StructsResponse.UserInfoStructResponse, 170
usernameTextField
FriendItemManager, 81
NoticeInviteManager, 114
UserOffline
Controllers, 52
UserOnline, 170
Controllers, 52
UseSkill
BaseEntity, 33
BaseSkill, 43
Drunkard, 64
Exclamation, 76
FireballSkill, 78
FlurryOfFire, 79
Toxicant, 161

value
Setting, 133
ValueChanged
SoundSliderManager, 146
volumeChange
SoundSettings, 144
Vuji/Assets/Scripts/Authorization/Login/LoginManager.cs, 173
Vuji/Assets/Scripts/Authorization/Register/RegisterManager.cs, 173
Vuji/Assets/Scripts/Controllers.cs, 174
Vuji/Assets/Scripts/DataBase.cs, 177
Vuji/Assets/Scripts/Game/AI/AgressionTrigger.cs, 180
Vuji/Assets/Scripts/Game/AI/AttackTrigger.cs, 180
Vuji/Assets/Scripts/Game/AI/EnemyAI.cs, 180
Vuji/Assets/Scripts/Game/AI/EntityMelee.cs, 181
Vuji/Assets/Scripts/Game/Attack/PlayerAOE.cs, 182

Vuji/Assets/Scripts/Game/Attack/PlayerMelee.cs, 182
Vuji/Assets/Scripts/Game/Attack/PlayerProjectile.cs, 184
Vuji/Assets/Scripts/Game/BaseObjects/BaseAOE.cs, 185
Vuji/Assets/Scripts/Game/BaseObjects/BaseEffect.cs, 186
Vuji/Assets/Scripts/Game/BaseObjects/BaseEntity.cs, 186
Vuji/Assets/Scripts/Game/BaseObjects/BaseItem.cs, 190
Vuji/Assets/Scripts/Game/BaseObjects/BasePassiveSkill.cs, 191
Vuji/Assets/Scripts/Game/BaseObjects/BaseProjectile.cs, 191
Vuji/Assets/Scripts/Game/BaseObjects/BaseSkill.cs, 193
Vuji/Assets/Scripts/Game/BaseObjects/BaseUpgrade.cs, 194
Vuji/Assets/Scripts/Game/Camera/CameraFollow.cs, 194
Vuji/Assets/Scripts/Game/Effects/Drunk.cs, 194
Vuji/Assets/Scripts/Game/Effects/OnFire.cs, 195
Vuji/Assets/Scripts/Game/Effects/Poison.cs, 195
Vuji/Assets/Scripts/Game/Effects/SpeedUp.cs, 195
Vuji/Assets/Scripts/Game/Effects/Strong.cs, 196
Vuji/Assets/Scripts/Game/EndGame.cs, 196
Vuji/Assets/Scripts/Game/Inventory/DisplayedItem.cs, 196
Vuji/Assets/Scripts/Game/Inventory/Inventory.cs, 197
Vuji/Assets/Scripts/Game/Inventory/InventoryWindow.cs, 198
Vuji/Assets/Scripts/Game/Items/Appe.cs, 199
Vuji/Assets/Scripts/Game/Items/DroppedItem.cs, 199
Vuji/Assets/Scripts/Game/ManagerGame.cs, 200
Vuji/Assets/Scripts/Game/Map/Generator.cs, 202
Vuji/Assets/Scripts/Game/Map/Room.cs, 203
Vuji/Assets/Scripts/Game/Map/StartEndZone.cs, 203
Vuji/Assets/Scripts/Game/PassiveSkills/Berserk.cs, 203
Vuji/Assets/Scripts/Game/Player/AnimationPlayer.cs, 204
Vuji/Assets/Scripts/Game/Player/CameraPlayer.cs, 205
Vuji/Assets/Scripts/Game/Player/ClassSelection.cs, 206
Vuji/Assets/Scripts/Game/Player/InputControllerPlayer.cs, 207
Vuji/Assets/Scripts/Game/Player/MovementPlayer.cs, 207
Vuji/Assets/Scripts/Game/Player/PlayerScript.cs, 209
Vuji/Assets/Scripts/Game/Player/PlayerUpgrades.cs, 209
Vuji/Assets/Scripts/Game/Skills/Drunkard.cs, 210
Vuji/Assets/Scripts/Game/Skills/Exclamation.cs, 211
Vuji/Assets/Scripts/Game/Skills/FireballSkill.cs, 211
Vuji/Assets/Scripts/Game/Skills/FlurryOfFire.cs, 212
Vuji/Assets/Scripts/Game/Skills/Toxicant.cs, 212
Vuji/Assets/Scripts/Game/Sounds/SoundManager.cs, 213
Vuji/Assets/Scripts/Game/Spawn/SpawnEnemy.cs, 213
Vuji/Assets/Scripts/Game/Spawn/SpawnPlayers.cs, 214
Vuji/Assets/Scripts/Game/Teams/PlayersTeamsManager.cs, 214
Vuji/Assets/Scripts/Loading/Connect.cs, 215
Vuji/Assets/Scripts/Lobby/AcceptFriendInvite.cs, 215
Vuji/Assets/Scripts/Lobby/FriendsListController.cs, 216
Vuji/Assets/Scripts/Lobby/InviteFriend.cs, 217
Vuji/Assets/Scripts/Lobby/LeaveRoom.cs, 217
Vuji/Assets/Scripts/Lobby/LobbyManager.cs, 217
Vuji/Assets/Scripts/Lobby/NoticeListController.cs, 219
Vuji/Assets/Scripts/Lobby/Play.cs, 219
Vuji/Assets/Scripts/Lobby/PlayersFounded.cs, 221
Vuji/Assets/Scripts/Lobby/SocketServerController.cs, 221
Vuji/Assets/Scripts/Lobby/StartGameLevel.cs, 223
Vuji/Assets/Scripts/Lobby/StopSearchGame.cs, 224
Vuji/Assets/Scripts/Structs.cs, 225
Vuji/Assets/Scripts/UIScripts/DISABLESUKA.cs, 226
Vuji/Assets/Scripts/UIScripts/ENABLESUKA.cs, 227
Vuji/Assets/Scripts/UIScripts/KeyHandler.cs, 227
Vuji/Assets/Scripts/UIScripts/Managers/EffectsListManager.cs, 229
Vuji/Assets/Scripts/UIScripts/Managers/EffectTimerManager.cs, 230
Vuji/Assets/Scripts/UIScripts/Managers/EndGameManager.cs, 231
Vuji/Assets/Scripts/UIScripts/Managers/EntityNameManager.cs, 231
Vuji/Assets/Scripts/UIScripts/Managers/EscapeMenu.cs, 232
Vuji/Assets/Scripts/UIScripts/Managers/FriendItemManager.cs, 232
Vuji/Assets/Scripts/UIScripts/Managers/HealthBarManager.cs, 233
Vuji/Assets/Scripts/UIScripts/Managers/KeybindManager.cs, 233

Vuji/Assets/Scripts/UIScripts/Managers/NoticeInviteManager.cs, 234
Vuji/Assets/Scripts/UIScripts/Managers/SettingsManager.cs, 235
Vuji/Assets/Scripts/UIScripts/Managers/SkillPanelManager.cs, 237
Vuji/Assets/Scripts/UIScripts/Managers/SoundSettings.cs, 238
Vuji/Assets/Scripts/UIScripts/Managers/SoundSliderManager.cs, 238
Vuji/Assets/Scripts/UIScripts/Managers/TeamHpPanelManager.cs, 239
Vuji/Assets/Scripts/UIScripts/Managers/TimerManager.cs, 239
Vuji/Assets/Scripts/UIScripts/Managers/TimerWithSpritemanager.cs, 240
Vuji/Assets/Scripts/UIScripts/Managers/UIHintManager.cs, 240
Vuji/Assets/Scripts/UIScripts/Units/EnergyBarUI.cs, 242
Vuji/Assets/Scripts/UIScripts/Units/HealthBarUI.cs, 243
Vuji/Assets/Scripts/UIScripts/Units/SkillCastTimer.cs, 243
Vuji/Assets/Scripts/UIScripts/Units/TeamHPBarUI.cs, 244
Vuji/Assets/Scripts/UIScripts/Units/TooltipText.cs, 244
Vuji/Assets/Scripts/UIScripts/Units/TooltipTextUI.cs, 245
Vuji/Assets/Scripts/UnityMainThreadDispatcher.cs, 245
Vuji/Assets/Scripts/UserOnline.cs, 246

Width
    Generator, 84