CardHouse

1.0.4

Generated by Doxygen 1.9.7

1 Namespace Index 1
1.1 Package List . . . 1

2 Hierarchical Index 3
2.1 Class Hierarchy . . . 3

3 Class Index 7
3.1 Class List . . . 7

4 File Index 11
4.1 File List . . . 11

5 Namespace Documentation 15
5.1 CardHouse Namespace Reference . . . 15
5.1.1 Enumeration Type Documentation . . . 17
5.1.1.1 CardFacing . . . 17
5.1.1.2 DragAction . . . 18
5.1.1.3 GroupInteractability . . . 18
5.1.1.4 GroupName . . . 18
5.1.1.5 GroupTargetType . . . 18
5.1.1.6 Loyalty . . . 18
5.1.1.7 MajorArcanaName . . . 18
5.1.1.8 MountingMode . . . 18
5.1.1.9 PokerSuit . . . 18
5.1.1.10 TarotSuit . . . 19
5.2 CardHouse.SampleGames Namespace Reference . . . 19
5.3 CardHouse.SampleGames.DeckBuilder Namespace Reference . . . 19
5.4 CardHouse.SampleGames.MemoryMatch Namespace Reference . . . 19
5.5 CardHouse.SampleGames.Solitaire Namespace Reference . . . 19
5.6 CardHouse.SampleGames.Tarot Namespace Reference . . . 19
5.7 CardHouse.TestScenes Namespace Reference . . . 20
5.8 CardHouse.Tutorial Namespace Reference . . . 20

6 Class Documentation 21
6.1 CardHouse.Activatable Class Reference . . . 21
6.1.1 Detailed Description . . . 21
6.1.2 Member Function Documentation . . . 21
6.1.2.1 Activate() . . . 21
6.1.2.2 OnActivate() . . . 22
6.2 CardHouse.AnimCurveFloatSeeker Class Reference . . . 22
6.2.1 Detailed Description . . . 23
6.2.2 Constructor & Destructor Documentation . . . 23
6.2.2.1 AnimCurveFloatSeeker() . . . 23
6.2.3 Member Function Documentation . . . 23

6.2.3.1 IsDone() . . . 23
6.2.3.2 MakeCopy() . . . 23
6.2.3.3 Pump() . . . 23
6.2.4 Member Data Documentation . . . 24
6.2.4.1 Duration . . . 24
6.2.4.2 ProgressCurve . . . 24
6.2.4.3 Timer . . . 24
6.3 CardHouse.AnimCurveFloatSeekerScriptable Class Reference . . . 24
6.3.1 Detailed Description . . . 25
6.3.2 Member Function Documentation . . . 25
6.3.2.1 GetStrategy() . . . 25
6.3.3 Member Data Documentation . . . 25
6.3.3.1 Duration . . . 25
6.3.3.2 ProgressCurve . . . 25
6.4 CardHouse.AnimCurveVector3Seeker Class Reference . . . 25
6.4.1 Detailed Description . . . 26
6.4.2 Constructor & Destructor Documentation . . . 26
6.4.2.1 AnimCurveVector3Seeker() . . . 26
6.4.3 Member Function Documentation . . . 27
6.4.3.1 IsDone() . . . 27
6.4.3.2 MakeCopy() . . . 27
6.4.3.3 Pump() . . . 27
6.4.4 Member Data Documentation . . . 27
6.4.4.1 Duration . . . 27
6.4.4.2 ProgressCurve . . . 27
6.4.4.3 Timer . . . 27
6.5 CardHouse.AnimCurveVector3SeekerScriptable Class Reference . . . 28
6.5.1 Detailed Description . . . 28
6.5.2 Member Function Documentation . . . 28
6.5.2.1 GetStrategy() . . . 28
6.5.3 Member Data Documentation . . . 28
6.5.3.1 Duration . . . 28
6.5.3.2 ProgressCurve . . . 29
6.6 CardHouse.ArcanaData Class Reference . . . 29
6.6.1 Detailed Description . . . 29
6.6.2 Member Data Documentation . . . 29
6.6.2.1 Arcana . . . 29
6.6.2.2 Rank . . . 29
6.6.2.3 Suit . . . 29
6.7 CardHouse.BaseSeekerComponent< T > Class Template Reference . . . 30
6.7.1 Detailed Description . . . 30
6.7.2 Member Function Documentation . . . 30

6.7.2.1 StartSeeking() . . . 30
6.7.3 Member Data Documentation . . . 31
6.7.3.1 IsSeeking . . . 31
6.7.3.2 MyStrategy . . . 31
6.7.3.3 Strategy . . . 31
6.7.3.4 UseLocalSpace . . . 31
6.8 CardHouse.BlockAllDrops Class Reference . . . 31
6.8.1 Detailed Description . . . 32
6.8.2 Member Function Documentation . . . 32
6.8.2.1 IsUnlockedInternal() . . . 32
6.9 CardHouse.Card Class Reference . . . 32
6.9.1 Detailed Description . . . 33
6.9.2 Member Function Documentation . . . 33
6.9.2.1 GetDiscardGroup() . . . 33
6.9.2.2 HandlePlayed() . . . 34
6.9.2.3 SetFacing() [1/2] . . . 34
6.9.2.4 SetFacing() [2/2] . . . 34
6.9.2.5 SetFocus() . . . 34
6.9.2.6 SetUpsideDown() . . . 34
6.9.2.7 ToggleFocus() . . . 34
6.9.2.8 TriggerMountEvents() . . . 34
6.9.2.9 TriggerUnMountEvents() . . . 35
6.9.3 Member Data Documentation . . . 35
6.9.3.1 CanBeUpsideDown . . . 35
6.9.3.2 FaceHoming . . . 35
6.9.3.3 FaceScaling . . . 35
6.9.3.4 FaceTurning . . . 35
6.9.3.5 FlipAnimator . . . 35
6.9.3.6 Group . . . 35
6.9.3.7 GroupTransitionEvents . . . 36
6.9.3.8 OnCardFocused . . . 36
6.9.3.9 OnFlipDown . . . 36
6.9.3.10 OnFlipUp . . . 36
6.9.3.11 OnMount . . . 36
6.9.3.12 OnPlay . . . 36
6.9.3.13 RootToRotateWhenUpsideDown . . . 36
6.9.3.14 UpsideDownChance . . . 36
6.9.4 Property Documentation . . . 37
6.9.4.1 Facing . . . 37
6.9.4.2 Homing . . . 37
6.9.4.3 IsUpsideDown . . . 37
6.9.4.4 Scaling . . . 37

6.9.4.5 Turning . . . 37
6.10 CardHouse.CardDefinition Class Reference . . . 37
6.10.1 Detailed Description . . . 38
6.10.2 Member Data Documentation . . . 38
6.10.2.1 BackArt . . . 38
6.11 CardHouse.Tutorial.CardDragTutorial Class Reference . . . 38
6.11.1 Detailed Description . . . 39
6.11.2 Member Function Documentation . . . 39
6.11.2.1 AdjustDragSwellSlider() . . . 39
6.11.2.2 AdjustOffsetX() . . . 39
6.11.2.3 AdjustOffsetY() . . . 39
6.11.2.4 AdjustSeekerGainSlider() . . . 39
6.11.2.5 OnGrabOffsetToggled() . . . 39
6.11.2.6 ShowSwellOutline() . . . 39
6.11.3 Member Data Documentation . . . 40
6.11.3.1 Card . . . 40
6.11.3.2 DragSwellSlider . . . 40
6.11.3.3 DragSwellText . . . 40
6.11.3.4 GrabOffsetToggle . . . 40
6.11.3.5 SeekerGainSlider . . . 40
6.11.3.6 SeekerGainText . . . 40
6.11.3.7 XOffsetSlider . . . 40
6.11.3.8 XOffsetText . . . 41
6.11.3.9 YOffsetSlider . . . 41
6.11.3.10 YOffsetText . . . 41
6.12 CardHouse.CardDropGateDimmer Class Reference . . . 41
6.12.1 Detailed Description . . . 42
6.12.2 Member Data Documentation . . . 42
6.12.2.1 ActiveMessage . . . 42
6.12.2.2 Handler . . . 42
6.12.2.3 InactiveMessage . . . 42
6.13 CardHouse.CardGridLayout Class Reference . . . 42
6.13.1 Detailed Description . . . 43
6.13.2 Member Function Documentation . . . 43
6.13.2.1 ApplySpacing() . . . 43
6.13.3 Member Data Documentation . . . 44
6.13.3.1 CardsPerRow . . . 44
6.13.3.2 MarginalCardOffset . . . 44
6.13.3.3 Straighten . . . 44
6.14 CardHouse.CardGroup Class Reference . . . 44
6.14.1 Detailed Description . . . 45
6.14.2 Member Function Documentation . . . 45

6.14.2.1 AddHoveredGroup() . . . 45
6.14.2.2 ApplyStrategy() . . . 45
6.14.2.3 Get() [1/2] . . . 46
6.14.2.4 Get() [2/2] . . . 46
6.14.2.5 GetActiveCard() . . . 46
6.14.2.6 GetClosestMountedCardIndex() . . . 46
6.14.2.7 HandleTriggerEnter2D() . . . 46
6.14.2.8 HandleTriggerExit2D() . . . 46
6.14.2.9 HasRoom() . . . 46
6.14.2.10 IndexOf() . . . 47
6.14.2.11 Mount() . . . 47
6.14.2.12 RemoveHoveredGroup() . . . 47
6.14.2.13 SafeMount() . . . 47
6.14.2.14 SetHilightState() . . . 47
6.14.2.15 Shuffle() . . . 47
6.14.2.16 ShuffleIn() . . . 48
6.14.2.17 UnMount() [1/2] . . . 48
6.14.2.18 UnMount() [2/2] . . . 48
6.14.3 Member Data Documentation . . . 48
6.14.3.1 DropGates . . . 48
6.14.3.2 Hilight . . . 48
6.14.3.3 HilightOnCardEntry . . . 48
6.14.3.4 MountedCards . . . 49
6.14.3.5 OnCardUsedOnTarget . . . 49
6.14.3.6 OnGroupChanged . . . 49
6.14.3.7 OnNewActiveGroup . . . 49
6.14.3.8 ShuffleStrategy . . . 49
6.14.4 Property Documentation . . . 49
6.14.4.1 HilightedGroup . . . 49
6.15 CardHouse.CardGroupSettings Class Reference . . . 50
6.15.1 Detailed Description . . . 50
6.15.2 Member Function Documentation . . . 50
6.15.2.1 Apply() . . . 50
6.15.3 Member Data Documentation . . . 50
6.15.3.1 CardLimit . . . 50
6.15.3.2 DragMountingMode . . . 51
6.15.3.3 ForcedFacing . . . 51
6.15.3.4 ForcedInteractability . . . 51
6.15.3.5 MountedCardAltitude . . . 51
6.15.3.6 UseMyScale . . . 51
6.16 CardHouse.CardLoyalty Class Reference . . . 51
6.16.1 Detailed Description . . . 52

6.16.2 Member Data Documentation . . . 52
6.16.2.1 PlayerIndex . . . 52
6.17 CardHouse.CardSetup Class Reference . . . 52
6.17.1 Detailed Description . . . 52
6.18 CardHouse.CardTargetCardOperator Class Reference . . . 53
6.18.1 Detailed Description . . . 53
6.18.2 Member Function Documentation . . . 53
6.18.2.1 OnActivate() . . . 53
6.18.3 Member Data Documentation . . . 54
6.18.3.1 DiscardSeekers . . . 54
6.18.3.2 MyCard . . . 54
6.18.3.3 Target . . . 54
6.19 CardHouse.CardTransferOperator Class Reference . . . 54
6.19.1 Detailed Description . . . 55
6.19.2 Member Function Documentation . . . 55
6.19.2.1 OnActivate() . . . 55
6.19.3 Member Data Documentation . . . 55
6.19.3.1 FlipSpeed . . . 55
6.19.3.2 GrabFrom . . . 55
6.19.3.3 NumberToTransfer . . . 55
6.19.3.4 OnSourceDepletedEventChain . . . 56
6.19.3.5 PopPushHomingOverride . . . 56
6.19.3.6 SendTo . . . 56
6.19.3.7 Transition . . . 56
6.19.3.8 TryAgainAfterSourceDepleted . . . 56
6.20 CardHouse.ClickDetector Class Reference . . . 56
6.20.1 Detailed Description . . . 57
6.20.2 Member Data Documentation . . . 57
6.20.2.1 ClickGates . . . 57
6.20.2.2 OnButtonClicked . . . 57
6.20.2.3 OnPress . . . 57
6.21 CardHouse.Tutorial.ClosestCardHighlighter Class Reference . . . 58
6.21.1 Detailed Description . . . 58
6.22 CardHouse.ContinuousInstantVector3Seeker Class Reference . . . 58
6.22.1 Detailed Description . . . 59
6.22.2 Member Function Documentation . . . 59
6.22.2.1 IsDone() . . . 59
6.22.2.2 MakeCopy() . . . 59
6.22.2.3 Pump() . . . 59
6.23 CardHouse.ContinuousInstantVector3SeekerScriptable Class Reference . . . 59
6.23.1 Detailed Description . . . 60
6.23.2 Member Function Documentation . . . 60

6.23.2.1 GetStrategy() . . . 60
6.24 CardHouse.CurrencyCost.CostWithLabel Class Reference . . . 60
6.24.1 Detailed Description . . . 60
6.24.2 Member Data Documentation . . . 60
6.24.2.1 Cost . . . 60
6.24.2.2 Label . . . 61
6.25 CardHouse.CurrencyChangeDetector Class Reference . . . 61
6.25.1 Detailed Description . . . 61
6.25.2 Member Data Documentation . . . 61
6.25.2.1 OnCurrencyChange . . . 61
6.26 CardHouse.CurrencyContainer Class Reference . . . 62
6.26.1 Detailed Description . . . 62
6.26.2 Member Function Documentation . . . 62
6.26.2.1 Adjust() . . . 62
6.26.2.2 Clone() . . . 63
6.26.3 Member Data Documentation . . . 63
6.26.3.1 HasMax . . . 63
6.26.3.2 HasMin . . . 63
6.26.3.3 Max . . . 63
6.26.3.4 Min . . . 63
6.26.3.5 RefillValue . . . 63
6.27 CardHouse.CurrencyCost Class Reference . . . 64
6.27.1 Detailed Description . . . 64
6.27.2 Member Function Documentation . . . 64
6.27.2.1 Activate() . . . 64
6.27.3 Member Data Documentation . . . 64
6.27.3.1 Cost . . . 64
6.28 CardHouse.CurrencyGate Class Reference . . . 65
6.28.1 Detailed Description . . . 65
6.28.2 Member Function Documentation . . . 65
6.28.2.1 IsUnlockedInternal() . . . 65
6.29 CardHouse.CurrencyOperator Class Reference . . . 66
6.29.1 Detailed Description . . . 66
6.29.2 Member Function Documentation . . . 66
6.29.2.1 Activate() . . . 66
6.29.3 Member Data Documentation . . . 66
6.29.3.1 MyRegistry . . . 66
6.30 CardHouse.CurrencyQuantity Class Reference . . . 67
6.30.1 Detailed Description . . . 67
6.30.2 Member Function Documentation . . . 67
6.30.2.1 Clone() . . . 67
6.30.3 Member Data Documentation . . . 67

6.30.3.1 Amount . . . 67
6.30.3.2 CurrencyType . . . 68
6.31 CardHouse.CurrencyRefillOperator Class Reference . . . 68
6.31.1 Detailed Description . . . 68
6.31.2 Member Function Documentation . . . 69
6.31.2.1 AdjustCurrencies() . . . 69
6.31.3 Member Data Documentation . . . 69
6.31.3.1 CurrenciesToRefill . . . 69
6.32 CardHouse.CurrencyRegistry Class Reference . . . 69
6.32.1 Detailed Description . . . 70
6.32.2 Member Function Documentation . . . 70
6.32.2.1 AdjustCurrency() . . . 70
6.32.2.2 GetCurrency() [1/2] . . . 70
6.32.2.3 GetCurrency() [2/2] . . . 70
6.32.2.4 Refill() . . . 70
6.32.3 Member Data Documentation . . . 71
6.32.3.1 CurrencyDisplayParent . . . 71
6.32.3.2 CurrencyDisplayPrefab . . . 71
6.32.3.3 CurrentPlayerLabel . . . 71
6.32.3.4 Instance . . . 71
6.32.3.5 OnCurrencyChanged . . . 71
6.32.3.6 PlayerWallets . . . 71
6.33 CardHouse.CurrencyScriptable Class Reference . . . 72
6.33.1 Detailed Description . . . 72
6.33.2 Member Data Documentation . . . 72
6.33.2.1 Name . . . 72
6.33.2.2 Sprite . . . 72
6.34 CardHouse.CurrencyUI Class Reference . . . 72
6.34.1 Detailed Description . . . 73
6.34.2 Member Function Documentation . . . 73
6.34.2.1 Apply() . . . 73
6.34.3 Member Data Documentation . . . 73
6.34.3.1 Image . . . 73
6.34.3.2 Text . . . 73
6.35 CardHouse.CurrencyWallet Class Reference . . . 73
6.35.1 Detailed Description . . . 74
6.35.2 Member Function Documentation . . . 74
6.35.2.1 CanAfford() . . . 74
6.35.2.2 Clone() . . . 74
6.35.2.3 FindCurrency() . . . 74
6.35.3 Member Data Documentation . . . 74
6.35.3.1 Currencies . . . 74

6.36 CardHouse.SampleGames.DeckBuilder.DamageGroupOperator Class Reference . . . 75
6.36.1 Detailed Description . . . 76
6.36.2 Member Function Documentation . . . 76
6.36.2.1 ActOnTarget() . . . 76
6.36.3 Member Data Documentation . . . 76
6.36.3.1 Damage . . . 76
6.37 CardHouse.SampleGames.DeckBuilder.DamageTargetOperator Class Reference . . . 76
6.37.1 Detailed Description . . . 77
6.37.2 Member Function Documentation . . . 77
6.37.2.1 ActOnTarget() . . . 77
6.37.3 Member Data Documentation . . . 78
6.37.3.1 Damage . . . 78
6.38 CardHouse.DeckDefinition Class Reference . . . 78
6.38.1 Detailed Description . . . 78
6.38.2 Member Data Documentation . . . 78
6.38.2.1 CardBackArt . . . 78
6.38.2.2 CardCollection . . . 78
6.39 CardHouse.DeckSetup Class Reference . . . 79
6.39.1 Detailed Description . . . 79
6.39.2 Member Function Documentation . . . 79
6.39.2.1 DoSetup() . . . 79
6.39.3 Member Data Documentation . . . 79
6.39.3.1 CardPrefab . . . 79
6.39.3.2 Deck . . . 79
6.39.3.3 DeckDefinition . . . 80
6.39.3.4 OnSetupCompleteEventChain . . . 80
6.39.3.5 RunOnStart . . . 80
6.40 CardHouse.Tutorial.DiscardAllCardsOperator Class Reference . . . 80
6.40.1 Detailed Description . . . 80
6.40.2 Member Function Documentation . . . 81
6.40.2.1 Activate() . . . 81
6.40.3 Member Data Documentation . . . 81
6.40.3.1 DiscardSeekers . . . 81
6.40.3.2 TargetDiscardSeekers . . . 81
6.41 CardHouse.DiscardCardOperator Class Reference . . . 81
6.41.1 Detailed Description . . . 81
6.41.2 Member Function Documentation . . . 82
6.41.2.1 Activate() . . . 82
6.42 CardHouse.DiscardTargetCardOperator Class Reference . . . 82
6.42.1 Detailed Description . . . 83
6.42.2 Member Function Documentation . . . 83
6.42.2.1 ActOnTarget() . . . 83

6.42.3 Member Data Documentation . . . 83
6.42.3.1 TargetDiscardSeekers . . . 83
6.43 CardHouse.DragDetector Class Reference . . . 83
6.43.1 Detailed Description . . . 84
6.43.2 Member Data Documentation . . . 84
6.43.2.1 DragGates . . . 84
6.43.2.2 GroupDropGates . . . 84
6.43.2.3 OnDragEnd . . . 84
6.43.2.4 OnDragStart . . . 84
6.43.2.5 TargetCardGates . . . 85
6.44 CardHouse.DragGateDimmer Class Reference . . . 85
6.44.1 Detailed Description . . . 85
6.44.2 Member Function Documentation . . . 86
6.44.2.1 UpdateHandler() . . . 86
6.44.3 Member Data Documentation . . . 86
6.44.3.1 ActiveMessage . . . 86
6.44.3.2 Handler . . . 86
6.44.3.3 InactiveMessage . . . 86
6.45 CardHouse.Dragging Class Reference . . . 86
6.45.1 Detailed Description . . . 87
6.45.2 Member Function Documentation . . . 87
6.45.2.1 BeginDragging() . . . 87
6.45.2.2 GetTarget() . . . 87
6.45.2.3 StopDragging() . . . 88
6.45.2.4 UpdateStrategy() . . . 88
6.45.3 Member Data Documentation . . . 88
6.45.3.1 CardPopupDistance . . . 88
6.45.3.2 DefaultCardZ . . . 88
6.45.3.3 DragHomingStrategy . . . 88
6.45.3.4 GrabOffset . . . 88
6.45.3.5 Instance . . . 88
6.45.3.6 OnDrag . . . 89
6.45.3.7 OnDrop . . . 89
6.45.3.8 PostDrop . . . 89
6.45.3.9 SetNewOffsetOnGrab . . . 89
6.45.3.10 UseGrabOffset . . . 89
6.46 CardHouse.DragOperator Class Reference . . . 89
6.46.1 Detailed Description . . . 90
6.46.2 Member Function Documentation . . . 90
6.46.2.1 SetDragState() . . . 90
6.46.3 Member Data Documentation . . . 90
6.46.3.1 DragAction . . . 90

6.46.3.2 DragSwell . . . 90
6.46.3.3 MyDragDetector . . . 90
6.46.3.4 PointUpWhenDragged . . . 91
6.46.3.5 PresentationSeekers . . . 91
6.47 CardHouse.DragTransition Class Reference . . . 91
6.47.1 Detailed Description . . . 91
6.47.2 Member Data Documentation . . . 91
6.47.2.1 DragAction . . . 91
6.48 CardHouse.DropParams Class Reference . . . 92
6.48.1 Detailed Description . . . 92
6.48.2 Member Data Documentation . . . 92
6.48.2.1 Card . . . 92
6.48.2.2 DragType . . . 92
6.48.2.3 Source . . . 92
6.48.2.4 Target . . . 92
6.49 CardHouse.EventChain Class Reference . . . 93
6.49.1 Detailed Description . . . 93
6.49.2 Member Function Documentation . . . 93
6.49.2.1 Activate() . . . 93
6.49.3 Member Data Documentation . . . 93
6.49.3.1 Events . . . 93
6.49.3.2 OnChainFinished . . . 93
6.50 CardHouse.Tutorial.EventChainsTutorial Class Reference . . . 94
6.50.1 Detailed Description . . . 94
6.50.2 Member Function Documentation . . . 94
6.50.2.1 StartTransition() . . . 94
6.50.3 Member Data Documentation . . . 94
6.50.3.1 Chaining . . . 94
6.50.3.2 Dropdown . . . 94
6.50.3.3 NoChaining . . . 95
6.50.3.4 SafeChaining . . . 95
6.51 CardHouse.ExponentialAngleFloatSeeker Class Reference . . . 95
6.51.1 Detailed Description . . . 96
6.51.2 Constructor & Destructor Documentation . . . 96
6.51.2.1 ExponentialAngleFloatSeeker() . . . 96
6.51.3 Member Function Documentation . . . 96
6.51.3.1 Pump() . . . 96
6.52 CardHouse.ExponentialAngleFloatSeekerScriptable Class Reference . . . 97
6.52.1 Detailed Description . . . 97
6.52.2 Member Function Documentation . . . 97
6.52.2.1 GetStrategy() . . . 97
6.52.3 Member Data Documentation . . . 97

6.52.3.1 ArrivalDistance . . . 97
6.52.3.2 Gain . . . 98
6.53 CardHouse.ExponentialFloatSeeker Class Reference . . . 98
6.53.1 Detailed Description . . . 99
6.53.2 Constructor & Destructor Documentation . . . 99
6.53.2.1 ExponentialFloatSeeker() . . . 99
6.53.3 Member Function Documentation . . . 99
6.53.3.1 IsDone() . . . 99
6.53.3.2 MakeCopy() . . . 99
6.53.3.3 Pump() . . . 99
6.53.4 Member Data Documentation . . . 100
6.53.4.1 ArrivalDistance . . . 100
6.53.4.2 Gain . . . 100
6.54 CardHouse.ExponentialFloatSeekerScriptable Class Reference . . . 100
6.54.1 Detailed Description . . . 100
6.54.2 Member Function Documentation . . . 101
6.54.2.1 GetStrategy() . . . 101
6.54.3 Member Data Documentation . . . 101
6.54.3.1 ArrivalDistance . . . 101
6.54.3.2 Gain . . . 101
6.55 CardHouse.ExponentialVector3Seeker Class Reference . . . 101
6.55.1 Detailed Description . . . 102
6.55.2 Constructor & Destructor Documentation . . . 102
6.55.2.1 ExponentialVector3Seeker() . . . 102
6.55.3 Member Function Documentation . . . 102
6.55.3.1 IsDone() . . . 102
6.55.3.2 MakeCopy() . . . 102
6.55.3.3 Pump() . . . 103
6.56 CardHouse.ExponentialVector3SeekerScriptable Class Reference . . . 103
6.56.1 Detailed Description . . . 103
6.56.2 Member Function Documentation . . . 103
6.56.2.1 GetStrategy() . . . 103
6.56.3 Member Data Documentation . . . 104
6.56.3.1 ArrivalDistance . . . 104
6.56.3.2 XYGain . . . 104
6.56.3.3 ZGain . . . 104
6.57 CardHouse.Gate< T > Class Template Reference . . . 104
6.57.1 Detailed Description . . . 105
6.57.2 Member Function Documentation . . . 105
6.57.2.1 IsUnlocked() . . . 105
6.58 CardHouse.GateCollection< T > Class Template Reference . . . 105
6.58.1 Detailed Description . . . 105

6.58.2 Member Function Documentation . . . 106
6.58.2.1 AllUnlocked() . . . 106
6.58.2.2 AnyUnlocked() . . . 106
6.58.3 Member Data Documentation . . . 106
6.58.3.1 Gates . . . 106
6.59 CardHouse.Tutorial.GridTutorial Class Reference . . . 106
6.59.1 Detailed Description . . . 107
6.59.2 Member Function Documentation . . . 107
6.59.2.1 AdjustCardLimit() . . . 107
6.59.2.2 AdjustCardsPerRow() . . . 107
6.59.2.3 AdjustXScale() . . . 107
6.59.2.4 AdjustYScale() . . . 107
6.59.3 Member Data Documentation . . . 108
6.59.3.1 CardLimitSlider . . . 108
6.59.3.2 CardLimitText . . . 108
6.59.3.3 CardsPerRowSlider . . . 108
6.59.3.4 CardsPerRowText . . . 108
6.59.3.5 Deck . . . 108
6.59.3.6 Grid . . . 108
6.59.3.7 XScaleSlider . . . 108
6.59.3.8 XScaleText . . . 109
6.59.3.9 YScaleSlider . . . 109
6.59.3.10 YScaleText . . . 109
6.60 CardHouse.GroupConditional Class Reference . . . 109
6.60.1 Detailed Description . . . 110
6.60.2 Member Function Documentation . . . 110
6.60.2.1 OnActivate() . . . 110
6.60.3 Member Data Documentation . . . 110
6.60.3.1 MyCard . . . 110
6.60.3.2 Responses . . . 110
6.61 CardHouse.GroupDropGateDimmer Class Reference . . . 110
6.61.1 Detailed Description . . . 111
6.61.2 Member Data Documentation . . . 111
6.61.2.1 ActiveMessage . . . 111
6.61.2.2 Handler . . . 111
6.61.2.3 InactiveMessage . . . 111
6.62 CardHouse.GroupNameUnityActionKvp Class Reference . . . 111
6.62.1 Detailed Description . . . 111
6.62.2 Member Data Documentation . . . 112
6.62.2.1 Key . . . 112
6.62.2.2 Value . . . 112
6.63 CardHouse.GroupSetup.GroupPopulationData Struct Reference . . . 112

6.63.1 Detailed Description . . . 112
6.63.2 Member Data Documentation . . . 112
6.63.2.1 CardCount . . . 112
6.63.2.2 CardPrefab . . . 112
6.63.2.3 Group . . . 113
6.64 CardHouse.GroupRegistry Class Reference . . . 113
6.64.1 Detailed Description . . . 113
6.64.2 Member Function Documentation . . . 114
6.64.2.1 Get() . . . 114
6.64.2.2 GetGroupName() . . . 114
6.64.2.3 GetLoyalty() . . . 114
6.64.2.4 GetOwnerIndex() . . . 114
6.64.3 Member Data Documentation . . . 114
6.64.3.1 Groups . . . 114
6.64.3.2 Instance . . . 114
6.65 CardHouse.GroupSetup Class Reference . . . 115
6.65.1 Detailed Description . . . 115
6.65.2 Member Function Documentation . . . 115
6.65.2.1 DoSetup() . . . 115
6.65.3 Member Data Documentation . . . 115
6.65.3.1 GroupPopulationList . . . 115
6.65.3.2 GroupsToShuffle . . . 116
6.65.3.3 OnSetupCompleteEventChain . . . 116
6.65.3.4 RunOnStart . . . 116
6.66 CardHouse.Tutorial.GroupSetupTutorial Class Reference . . . 116
6.66.1 Detailed Description . . . 117
6.66.2 Member Function Documentation . . . 117
6.66.2.1 AdjustASpadesSlider() . . . 117
6.66.2.2 AdjustHearts10Slider() . . . 117
6.66.2.3 AdjustQDiamondsSlider() . . . 117
6.66.2.4 AdjustShuffle() . . . 117
6.66.2.5 Setup() . . . 117
6.66.3 Member Data Documentation . . . 117
6.66.3.1 ASpadesSlider . . . 117
6.66.3.2 ASpadesText . . . 118
6.66.3.3 Deck . . . 118
6.66.3.4 Hearts10Slider . . . 118
6.66.3.5 Hearts10Text . . . 118
6.66.3.6 PullBackOperator . . . 118
6.66.3.7 QDiamondsSlider . . . 118
6.66.3.8 QDiamondsText . . . 118
6.66.3.9 SetupComponent . . . 118

6.66.3.10 ShuffleToggle . . . 119
6.67 CardHouse.GroupTransition Class Reference . . . 119
6.67.1 Detailed Description . . . 119
6.67.2 Member Data Documentation . . . 119
6.67.2.1 Destination . . . 119
6.67.2.2 Source . . . 119
6.68 CardHouse.MultiplayerBoardSetup.GroupTransitionByName Class Reference . . . 120
6.68.1 Detailed Description . . . 120
6.68.2 Member Data Documentation . . . 120
6.68.2.1 Destination . . . 120
6.68.2.2 DragAction . . . 120
6.68.2.3 Mode . . . 120
6.68.2.4 PhaseIndex . . . 120
6.68.2.5 Source . . . 121
6.69 CardHouse.Card.GroupTransitionEvent Class Reference . . . 121
6.69.1 Detailed Description . . . 121
6.69.2 Member Data Documentation . . . 121
6.69.2.1 EntryEvent . . . 121
6.69.2.2 ExitEvent . . . 121
6.69.2.3 Group . . . 121
6.70 CardHouse.SampleGames.DeckBuilder.Health Class Reference . . . 122
6.70.1 Detailed Description . . . 122
6.70.2 Member Function Documentation . . . 122
6.70.2.1 Change() . . . 122
6.70.3 Member Data Documentation . . . 122
6.70.3.1 HealthLevel . . . 122
6.70.3.2 HealthText . . . 122
6.70.3.3 OnDeath . . . 123
6.71 CardHouse.Homing Class Reference . . . 123
6.71.1 Detailed Description . . . 124
6.71.2 Member Function Documentation . . . 124
6.71.2.1 GetCurrentValue() . . . 124
6.71.2.2 GetDefaultSeeker() . . . 124
6.71.2.3 SetNewValue() . . . 124
6.72 CardHouse.HoverDetector Class Reference . . . 124
6.72.1 Detailed Description . . . 125
6.72.2 Member Data Documentation . . . 125
6.72.2.1 OnHover . . . 125
6.72.2.2 OnUnHover . . . 125
6.73 CardHouse.IncrementCurrencyOperator Class Reference . . . 125
6.73.1 Detailed Description . . . 126
6.73.2 Member Function Documentation . . . 126

6.73.2.1 AdjustCurrencies() . . . 126
6.73.3 Member Data Documentation . . . 126
6.73.3.1 CurrenciesToChange . . . 126
6.74 CardHouse.InstantFloatSeeker Class Reference . . . 127
6.74.1 Detailed Description . . . 127
6.74.2 Member Function Documentation . . . 127
6.74.2.1 IsDone() . . . 127
6.74.2.2 MakeCopy() . . . 128
6.74.2.3 Pump() . . . 128
6.75 CardHouse.InstantVector3Seeker Class Reference . . . 128
6.75.1 Detailed Description . . . 129
6.75.2 Member Function Documentation . . . 129
6.75.2.1 IsDone() . . . 129
6.75.2.2 MakeCopy() . . . 129
6.75.2.3 Pump() . . . 129
6.76 CardHouse.Tutorial.MultiBoardTutorial.InstructionImagePair Class Reference . . . 129
6.76.1 Detailed Description . . . 130
6.76.2 Member Data Documentation . . . 130
6.76.2.1 Image . . . 130
6.76.2.2 Text . . . 130
6.77 CardHouse.Tutorial.LaunchDataScriptable Class Reference . . . 130
6.77.1 Detailed Description . . . 130
6.77.2 Member Data Documentation . . . 131
6.77.2.1 ActiveScene . . . 131
6.77.2.2 LaunchedTutorial . . . 131
6.77.2.3 OpenScenes . . . 131
6.78 CardHouse.LifetimeDestructor Class Reference . . . 131
6.78.1 Detailed Description . . . 131
6.78.2 Member Data Documentation . . . 132
6.78.2.1 Lifetime . . . 132
6.79 CardHouse.LoyaltyGateCardDrop Class Reference . . . 132
6.79.1 Detailed Description . . . 132
6.79.2 Member Function Documentation . . . 133
6.79.2.1 IsUnlockedInternal() . . . 133
6.79.3 Member Data Documentation . . . 133
6.79.3.1 Destinations . . . 133
6.79.3.2 Loyalty . . . 133
6.80 CardHouse.Tutorial.MatureCropDragGate Class Reference . . . 133
6.80.1 Detailed Description . . . 134
6.80.2 Member Function Documentation . . . 134
6.80.2.1 IsUnlockedInternal() . . . 134
6.81 CardHouse.SampleGames.MemoryMatch.MemoryCard Class Reference . . . 134

6.81.1 Detailed Description . . . 135
6.81.2 Member Function Documentation . . . 135
6.81.2.1 Apply() . . . 135
6.81.2.2 OnFlippedUp() . . . 135
6.81.3 Member Data Documentation . . . 135
6.81.3.1 MySprite . . . 135
6.81.3.2 MySpriteRenderer . . . 135
6.82 CardHouse.SampleGames.MemoryMatch.MemoryGame Class Reference . . . 135
6.82.1 Detailed Description . . . 136
6.82.2 Member Function Documentation . . . 136
6.82.2.1 Flip() . . . 136
6.82.2.2 Restart() . . . 136
6.82.3 Member Data Documentation . . . 136
6.82.3.1 CardPrefab . . . 136
6.82.3.2 Instance . . . 137
6.82.3.3 MatchEffect . . . 137
6.82.3.4 MyUI . . . 137
6.82.3.5 Slots . . . 137
6.82.3.6 Sprites . . . 137
6.83 CardHouse.SampleGames.MemoryMatch.MemoryUI Class Reference . . . 137
6.83.1 Detailed Description . . . 138
6.83.2 Member Function Documentation . . . 138
6.83.2.1 UpdateMatches() . . . 138
6.83.2.2 UpdateTimer() . . . 138
6.83.3 Member Data Documentation . . . 138
6.83.3.1 MatchText . . . 138
6.83.3.2 TimerText . . . 138
6.84 CardHouse.MountDetector Class Reference . . . 139
6.84.1 Detailed Description . . . 139
6.84.2 Member Data Documentation . . . 139
6.84.2.1 OnMount . . . 139
6.85 CardHouse.Tutorial.MultiBoardTutorial Class Reference . . . 139
6.85.1 Detailed Description . . . 140
6.85.2 Member Function Documentation . . . 140
6.85.2.1 InstructionsBackward() . . . 140
6.85.2.2 InstructionsForward() . . . 140
6.85.2.3 SetupBoard() . . . 140
6.85.2.4 UpdatePlayerCount() . . . 141
6.85.2.5 UpdateSpacingMultiplier() . . . 141
6.85.3 Member Data Documentation . . . 141
6.85.3.1 BackButton . . . 141
6.85.3.2 CommonArea . . . 141

6.85.3.3 ForwardButton . . . 141
6.85.3.4 InstructionIndex . . . 141
6.85.3.5 Instructions . . . 141
6.85.3.6 InstructionsImage . . . 142
6.85.3.7 InstructionsRoot . . . 142
6.85.3.8 InstructionsText . . . 142
6.85.3.9 PageNumberText . . . 142
6.85.3.10 PlayerCountLabel . . . 142
6.85.3.11 PlayerCountSlider . . . 142
6.85.3.12 SetupButton . . . 142
6.85.3.13 SetupScript . . . 142
6.85.3.14 SpacingLabel . . . 143
6.85.3.15 SpacingSlider . . . 143
6.86 CardHouse.MultiplayerBoardSetup Class Reference . . . 143
6.86.1 Detailed Description . . . 144
6.86.2 Member Enumeration Documentation . . . 144
6.86.2.1 PvpMode . . . 144
6.86.3 Member Function Documentation . . . 144
6.86.3.1 GetAllBoards() . . . 144
6.86.3.2 Setup() . . . 144
6.86.4 Member Data Documentation . . . 144
6.86.4.1 PlayerBoard . . . 144
6.86.4.2 PlayerCount . . . 144
6.86.4.3 PlayerToPlayerInteractions . . . 144
6.86.4.4 RunOnStart . . . 145
6.86.4.5 SpacingMultiplier . . . 145
6.87 CardHouse.MultiSpriteOperator Class Reference . . . 145
6.87.1 Detailed Description . . . 145
6.87.2 Member Function Documentation . . . 145
6.87.2.1 Activate() [1/2] . . . 145
6.87.2.2 Activate() [2/2] . . . 146
6.87.2.3 Remove() . . . 146
6.87.2.4 RemoveVote() . . . 146
6.87.3 Member Data Documentation . . . 146
6.87.3.1 SpriteOperators . . . 146
6.88 CardHouse.SpriteColorOperator.NamedColor Class Reference . . . 146
6.88.1 Detailed Description . . . 146
6.88.2 Member Data Documentation . . . 147
6.88.2.1 Color . . . 147
6.88.2.2 Name . . . 147
6.89 CardHouse.GroupRegistry.NamedGroup Class Reference . . . 147
6.89.1 Detailed Description . . . 147

6.89.2 Member Data Documentation . . . 147
6.89.2.1 Group . . . 147
6.89.2.2 Name . . . 147
6.89.2.3 PlayerIndex . . . 148
6.90 CardHouse.SpriteImageOperator.NamedSprite Class Reference . . . 148
6.90.1 Detailed Description . . . 148
6.90.2 Member Data Documentation . . . 148
6.90.2.1 Name . . . 148
6.90.2.2 Sprite . . . 148
6.91 CardHouse.NoParams Class Reference . . . 148
6.91.1 Detailed Description . . . 148
6.92 CardHouse.Tutorial.OutLinks Class Reference . . . 149
6.92.1 Detailed Description . . . 149
6.92.2 Member Function Documentation . . . 149
6.92.2.1 OnPointerClick() . . . 149
6.92.3 Member Data Documentation . . . 149
6.92.3.1 Text . . . 149
6.93 CardHouse.Phase Class Reference . . . 150
6.93.1 Detailed Description . . . 150
6.93.2 Member Function Documentation . . . 150
6.93.2.1 End() . . . 150
6.93.2.2 IsValidDrag() . . . 150
6.93.2.3 IsValidDragStart() . . . 150
6.93.2.4 Start() . . . 151
6.93.3 Member Data Documentation . . . 151
6.93.3.1 ActiveButtons . . . 151
6.93.3.2 CameraPosition . . . 151
6.93.3.3 CardPresentationPosition . . . 151
6.93.3.4 Name . . . 151
6.93.3.5 OnPhaseEndEventChain . . . 151
6.93.3.6 OnPhaseStartEventChain . . . 151
6.93.3.7 PlayerIndex . . . 152
6.93.3.8 ValidClickTargets . . . 152
6.93.3.9 ValidDrags . . . 152
6.94 CardHouse.PhaseChangeDetector Class Reference . . . 152
6.94.1 Detailed Description . . . 152
6.94.2 Member Data Documentation . . . 153
6.94.2.1 OnPhaseChange . . . 153
6.95 CardHouse.PhaseConditional Class Reference . . . 153
6.95.1 Detailed Description . . . 153
6.95.2 Member Function Documentation . . . 154
6.95.2.1 OnActivate() . . . 154

6.95.3 Member Data Documentation . . . 154
6.95.3.1 Responses . . . 154
6.96 CardHouse.PhaseGateCardDragStart Class Reference . . . 154
6.96.1 Detailed Description . . . 155
6.96.2 Member Function Documentation . . . 155
6.96.2.1 IsUnlockedInternal() . . . 155
6.97 CardHouse.PhaseGateCardDrop Class Reference . . . 155
6.97.1 Detailed Description . . . 155
6.97.2 Member Function Documentation . . . 156
6.97.2.1 IsUnlockedInternal() . . . 156
6.98 CardHouse.PhaseGateClick Class Reference . . . 156
6.98.1 Detailed Description . . . 156
6.98.2 Member Function Documentation . . . 157
6.98.2.1 IsUnlockedInternal() . . . 157
6.99 CardHouse.Tutorial.PhaseLabelUpdater Class Reference . . . 157
6.99.1 Detailed Description . . . 157
6.99.2 Member Function Documentation . . . 157
6.99.2.1 UpdatePhaseLabel() . . . 157
6.99.3 Member Data Documentation . . . 158
6.99.3.1 PhaseText . . . 158
6.100 CardHouse.PhaseManager Class Reference . . . 158
6.100.1 Detailed Description . . . 159
6.100.2 Member Function Documentation . . . 159
6.100.2.1 HardReset() . . . 159
6.100.2.2 IsValidClick() . . . 159
6.100.2.3 IsValidDrag() . . . 159
6.100.2.4 IsValidDragStart() . . . 159
6.100.2.5 NextPhase() . . . 159
6.100.2.6 SetCameraPosition() . . . 159
6.100.3 Member Data Documentation . . . 160
6.100.3.1 AllPhaseDependentButtons . . . 160
6.100.3.2 Instance . . . 160
6.100.3.3 OnPhaseChanged . . . 160
6.100.3.4 Phases . . . 160
6.100.4 Property Documentation . . . 160
6.100.4.1 CurrentPhase . . . 160
6.100.4.2 PlayerIndex . . . 160
6.101 CardHouse.Tutorial.Plant Class Reference . . . 161
6.101.1 Detailed Description . . . 161
6.101.2 Member Function Documentation . . . 161
6.101.2.1 CanBeWatered() . . . 161
6.101.2.2 HideCost() . . . 161

6.101.2.3 Payoff() . . . 162
6.101.2.4 Water() . . . 162
6.101.3 Member Data Documentation . . . 162
6.101.3.1 CostJewel . . . 162
6.101.3.2 CostText . . . 162
6.101.3.3 DescriptionText . . . 162
6.101.3.4 NameText . . . 162
6.101.3.5 PossiblePlants . . . 162
6.101.3.6 Sprite . . . 163
6.101.3.7 Value . . . 163
6.102 CardHouse.Tutorial.PlantGrowthScriptable Class Reference . . . 163
6.102.1 Detailed Description . . . 163
6.102.2 Member Data Documentation . . . 163
6.102.2.1 Stages . . . 163
6.103 CardHouse.Tutorial.PlantMaturityInfo Class Reference . . . 164
6.103.1 Detailed Description . . . 164
6.103.2 Member Data Documentation . . . 164
6.103.2.1 Description . . . 164
6.103.2.2 Name . . . 164
6.103.2.3 Sprite . . . 164
6.104 CardHouse.PokerCard Class Reference . . . 164
6.104.1 Detailed Description . . . 165
6.104.2 Member Function Documentation . . . 165
6.104.2.1 Apply() . . . 165
6.104.3 Member Data Documentation . . . 165
6.104.3.1 BackImage . . . 165
6.104.3.2 Image . . . 165
6.104.4 Property Documentation . . . 166
6.104.4.1 Rank . . . 166
6.104.4.2 Suit . . . 166
6.105 CardHouse.PokerCardDefinition Class Reference . . . 166
6.105.1 Detailed Description . . . 166
6.105.2 Member Data Documentation . . . 167
6.105.2.1 Art . . . 167
6.105.2.2 Rank . . . 167
6.105.2.3 Suit . . . 167
6.106 CardHouse.Tutorial.PresentationPointTutorial Class Reference . . . 167
6.106.1 Detailed Description . . . 168
6.106.2 Member Function Documentation . . . 168
6.106.2.1 Rotate() . . . 168
6.106.2.2 Scale() . . . 168
6.106.2.3 UpdateCameraPosition() . . . 168

6.106.3 Member Data Documentation . . . 168
6.106.3.1 ButtonParentScaling . . . 168
6.106.3.2 ButtonParentTurning . . . 168
6.106.3.3 ParentScaling . . . 168
6.106.3.4 ParentTurning . . . 169
6.106.3.5 PlayerIndex . . . 169
6.106.3.6 RotationMax . . . 169
6.106.3.7 RotationMin . . . 169
6.106.3.8 RotationSteps . . . 169
6.106.3.9 ScaleMax . . . 169
6.106.3.10 ScaleMin . . . 169
6.106.3.11 ScaleSteps . . . 170
6.107 CardHouse.RandomizedCurveVector3SeekerScriptable Class Reference . . . 170
6.107.1 Detailed Description . . . 171
6.107.2 Member Function Documentation . . . 171
6.107.2.1 GetStrategy() . . . 171
6.107.3 Member Data Documentation . . . 171
6.107.3.1 TweakMagnitudeMax . . . 171
6.107.3.2 TweakMagnitudeMin . . . 171
6.108 CardHouse.Tutorial.SandboxManager Class Reference . . . 171
6.108.1 Detailed Description . . . 172
6.108.2 Member Function Documentation . . . 172
6.108.2.1 GoTo() . . . 172
6.108.2.2 GoToNext() . . . 172
6.108.2.3 GoToPrevious() . . . 172
6.108.2.4 Reset() . . . 173
6.108.2.5 Start() . . . 173
6.108.2.6 ToggleSidebar() . . . 173
6.108.3 Member Data Documentation . . . 173
6.108.3.1 MultiBoardTutorial . . . 173
6.108.3.2 NextButton . . . 173
6.108.3.3 PreviousButton . . . 173
6.108.3.4 ResetButton . . . 173
6.108.3.5 SidebarAnimator . . . 174
6.108.3.6 TitleText . . . 174
6.108.3.7 TutorialButtonPrefab . . . 174
6.108.3.8 TutorialListRoot . . . 174
6.108.3.9 Tutorials . . . 174
6.109 CardHouse.Scaling Class Reference . . . 174
6.109.1 Detailed Description . . . 175
6.109.2 Member Function Documentation . . . 175
6.109.2.1 GetCurrentValue() . . . 175

6.109.2.2 GetDefaultSeeker() . . . 175
6.109.2.3 SetNewValue() . . . 176
6.110 CardHouse.Tutorial.SceneKeeper Class Reference . . . 176
6.110.1 Detailed Description . . . 176
6.111 CardHouse.Tutorial.SceneSpawner Class Reference . . . 176
6.111.1 Detailed Description . . . 176
6.111.2 Member Data Documentation . . . 177
6.111.2.1 SceneToSpawn . . . 177
6.112 CardHouse.Seeker< T > Class Template Reference . . . 177
6.112.1 Detailed Description . . . 177
6.112.2 Member Function Documentation . . . 177
6.112.2.1 StartSeeking() . . . 177
6.112.3 Member Data Documentation . . . 178
6.112.3.1 End . . . 178
6.112.3.2 Start . . . 178
6.113 CardHouse.SeekerScriptable< T > Class Template Reference . . . 178
6.113.1 Detailed Description . . . 178
6.114 CardHouse.SeekerScriptableSet Class Reference . . . 178
6.114.1 Detailed Description . . . 179
6.114.2 Member Data Documentation . . . 179
6.114.2.1 Homing . . . 179
6.114.2.2 Scaling . . . 179
6.114.2.3 Turning . . . 179
6.115 CardHouse.SeekerSet Class Reference . . . 179
6.115.1 Detailed Description . . . 179
6.115.2 Member Data Documentation . . . 180
6.115.2.1 Card . . . 180
6.115.2.2 FlipSpeed . . . 180
6.115.2.3 Homing . . . 180
6.115.2.4 Scaling . . . 180
6.115.2.5 Turning . . . 180
6.116 CardHouse.SeekerSetList Class Reference . . . 180
6.116.1 Detailed Description . . . 181
6.116.2 Member Function Documentation . . . 181
6.116.2.1 GetSeekerSetFor() . . . 181
6.117 CardHouse.Tutorial.SeekerTutorial Class Reference . . . 181
6.117.1 Detailed Description . . . 182
6.117.2 Member Function Documentation . . . 182
6.117.2.1 Transfer() . . . 182
6.117.3 Member Data Documentation . . . 182
6.117.3.1 HomingDropdown . . . 182
6.117.3.2 ScalingDropdown . . . 182

6.117.3.3 SeekerKVPs . . . 182
6.117.3.4 Stacks . . . 182
6.117.3.5 TurningDropdown . . . 182
6.117.3.6 Waypoint . . . 183
6.118 CardHouse.ShuffleOperator Class Reference . . . 183
6.118.1 Detailed Description . . . 183
6.118.2 Member Function Documentation . . . 184
6.118.2.1 OnActivate() . . . 184
6.118.3 Member Data Documentation . . . 184
6.118.3.1 Deck . . . 184
6.118.3.2 GroupsToShuffleIntoDeck . . . 184
6.119 CardHouse.SlotLayout Class Reference . . . 184
6.119.1 Detailed Description . . . 185
6.119.2 Member Function Documentation . . . 185
6.119.2.1 ApplySpacing() . . . 185
6.120 CardHouse.SampleGames.Solitaire.SolitaireCardDragHandler Class Reference . . . 185
6.120.1 Detailed Description . . . 186
6.120.2 Member Function Documentation . . . 186
6.120.2.1 AttachChildren() . . . 186
6.120.2.2 DetatchChildren() . . . 186
6.121 CardHouse.SampleGames.Solitaire.SolitaireColumnChangeHandler Class Reference . . . 186
6.121.1 Detailed Description . . . 186
6.121.2 Member Function Documentation . . . 187
6.121.2.1 Refresh() . . . 187
6.122 CardHouse.SampleGames.Solitaire.SolitaireColumnDropGate Class Reference . . . 187
6.122.1 Detailed Description . . . 187
6.122.2 Member Function Documentation . . . 188
6.122.2.1 IsUnlockedInternal() . . . 188
6.123 CardHouse.SampleGames.Solitaire.SolitaireDeckClickHandler Class Reference . . . 188
6.123.1 Detailed Description . . . 188
6.123.2 Member Function Documentation . . . 188
6.123.2.1 FlipOrReset() . . . 188
6.123.3 Member Data Documentation . . . 189
6.123.3.1 DealCardHandler . . . 189
6.123.3.2 FlipHandler . . . 189
6.123.3.3 MoveToDeckHandler . . . 189
6.123.3.4 ResetEventChain . . . 189
6.123.3.5 ShuffleHandler . . . 189
6.124 CardHouse.SampleGames.Solitaire.SolitaireScorePileDropGate Class Reference . . . 189
6.124.1 Detailed Description . . . 190
6.124.2 Member Function Documentation . . . 190
6.124.2.1 IsUnlockedInternal() . . . 190

6.125 CardHouse.SampleGames.Solitaire.SolitaireSetup Class Reference . . . 190
6.125.1 Detailed Description . . . 191
6.125.2 Member Function Documentation . . . 191
6.125.2.1 AllowReset() . . . 191
6.125.2.2 DealCards() . . . 191
6.125.2.3 PreventReset() . . . 191
6.125.2.4 TryResetBoard() . . . 191
6.125.3 Member Data Documentation . . . 192
6.125.3.1 AllGroups . . . 192
6.125.3.2 Columns . . . 192
6.125.3.3 DealingStrategy . . . 192
6.125.3.4 Deck . . . 192
6.125.3.5 ResetBoardEventChain . . . 192
6.126 CardHouse.SplayLayout Class Reference . . . 192
6.126.1 Detailed Description . . . 193
6.126.2 Member Function Documentation . . . 193
6.126.2.1 ApplySpacing() . . . 193
6.126.3 Member Data Documentation . . . 194
6.126.3.1 ArcCenterOffset . . . 194
6.126.3.2 ArcMargin . . . 194
6.126.3.3 MarginalCardOffset . . . 194
6.127 CardHouse.Tutorial.SplayTutorial Class Reference . . . 194
6.127.1 Detailed Description . . . 195
6.127.2 Member Function Documentation . . . 195
6.127.2.1 AdjustArcMargin() . . . 195
6.127.2.2 AdjustXOffset() . . . 195
6.127.2.3 AdjustXScale() . . . 195
6.127.2.4 AdjustYOffset() . . . 195
6.127.3 Member Data Documentation . . . 196
6.127.3.1 ArcMarginSlider . . . 196
6.127.3.2 ArcMarginText . . . 196
6.127.3.3 Deck . . . 196
6.127.3.4 Reticle . . . 196
6.127.3.5 Splay . . . 196
6.127.3.6 XOffsetSlider . . . 196
6.127.3.7 XOffsetText . . . 196
6.127.3.8 XScaleSlider . . . 197
6.127.3.9 XScaleText . . . 197
6.127.3.10 YOffsetSlider . . . 197
6.127.3.11 YOffsetText . . . 197
6.128 CardHouse.SampleGames.Tarot.SpreadManager Class Reference . . . 197
6.128.1 Detailed Description . . . 198

6.128.2 Member Function Documentation . . . 198
6.128.2.1 DealNextCard() . . . 198
6.128.2.2 NextSpread() . . . 198
6.128.2.3 PreviousSpread() . . . 198
6.128.2.4 ShuffleCardsBackIn() . . . 198
6.128.3 Member Data Documentation . . . 198
6.128.3.1 Deck . . . 198
6.128.3.2 Key . . . 199
6.128.3.3 SpreadLabel . . . 199
6.128.3.4 SpreadOrderLabelPrefab . . . 199
6.128.3.5 Spreads . . . 199
6.129 CardHouse.SpriteColorOperator Class Reference . . . 199
6.129.1 Detailed Description . . . 200
6.129.2 Member Function Documentation . . . 200
6.129.2.1 ChangeSprite() . . . 200
6.129.3 Member Data Documentation . . . 201
6.129.3.1 Colors . . . 201
6.130 CardHouse.SpriteImageOperator Class Reference . . . 201
6.130.1 Detailed Description . . . 202
6.130.2 Member Function Documentation . . . 202
6.130.2.1 ChangeSprite() . . . 202
6.130.3 Member Data Documentation . . . 202
6.130.3.1 Sprites . . . 202
6.131 CardHouse.SpriteOperator Class Reference . . . 202
6.131.1 Detailed Description . . . 203
6.131.2 Member Function Documentation . . . 203
6.131.2.1 Activate() [1/2] . . . 203
6.131.2.2 Activate() [2/2] . . . 203
6.131.2.3 Remove() . . . 203
6.131.3 Member Data Documentation . . . 204
6.131.3.1 FavoredState . . . 204
6.131.3.2 SpriteTarget . . . 204
6.132 CardHouse.Tutorial.SpriteOperatorTutorial Class Reference . . . 204
6.132.1 Detailed Description . . . 204
6.132.2 Member Function Documentation . . . 205
6.132.2.1 RegisterColorVote() . . . 205
6.132.2.2 RegisterImageVote() . . . 205
6.132.2.3 RemoveColorVote() . . . 205
6.132.2.4 RemoveImageVote() . . . 205
6.132.3 Member Data Documentation . . . 205
6.132.3.1 ColorOperator . . . 205
6.132.3.2 ImageOperator . . . 205

6.132.3.3 Instance . . . 206
6.133 CardHouse.Tutorial.SpriteVoterTutorial Class Reference . . . 206
6.133.1 Detailed Description . . . 206
6.133.2 Member Function Documentation . . . 206
6.133.2.1 OnColorDropdownUpdated() . . . 206
6.133.2.2 OnImageDropdownUpdated() . . . 206
6.133.3 Member Data Documentation . . . 207
6.133.3.1 OnStart . . . 207
6.134 CardHouse.StackLayout Class Reference . . . 207
6.134.1 Detailed Description . . . 208
6.134.2 Member Function Documentation . . . 208
6.134.2.1 ApplySpacing() . . . 208
6.134.3 Member Data Documentation . . . 208
6.134.3.1 MarginalCardOffset . . . 208
6.134.3.2 SecondaryCollider . . . 208
6.134.3.3 Straighten . . . 208
6.135 CardHouse.Tutorial.StackTutorial Class Reference . . . 209
6.135.1 Detailed Description . . . 209
6.135.2 Member Function Documentation . . . 209
6.135.2.1 AdjustXOffset() . . . 209
6.135.2.2 AdjustYOffset() . . . 209
6.135.2.3 UseColumnPreset() . . . 210
6.135.2.4 UseCompactDeckPreset() . . . 210
6.135.2.5 UseDeckPreset() . . . 210
6.135.2.6 UseRowPreset() . . . 210
6.135.3 Member Data Documentation . . . 210
6.135.3.1 Stack . . . 210
6.135.3.2 XOffsetSlider . . . 210
6.135.3.3 XOffsetText . . . 210
6.135.3.4 YOffsetSlider . . . 211
6.135.3.5 YOffsetText . . . 211
6.136 CardHouse.Tutorial.StringListScriptable Class Reference . . . 211
6.136.1 Detailed Description . . . 211
6.136.2 Member Data Documentation . . . 211
6.136.2.1 MyList . . . 211
6.137 CardHouse.Tutorial.StringSeekerKVP Class Reference . . . 212
6.137.1 Detailed Description . . . 212
6.137.2 Member Data Documentation . . . 212
6.137.2.1 Key . . . 212
6.137.2.2 Value . . . 212
6.138 CardHouse.StringUnityActionKvp Class Reference . . . 212
6.138.1 Detailed Description . . . 212

6.138.2 Member Data Documentation . . . 212
6.138.2.1 Key . . . 212
6.138.2.2 Value . . . 213
6.139 CardHouse.TargetCardParams Class Reference . . . 213
6.139.1 Detailed Description . . . 213
6.139.2 Member Data Documentation . . . 213
6.139.2.1 Source . . . 213
6.139.2.2 Target . . . 213
6.140 CardHouse.TarotCard Class Reference . . . 213
6.140.1 Detailed Description . . . 214
6.140.2 Member Enumeration Documentation . . . 214
6.140.2.1 Arcana . . . 214
6.140.3 Member Function Documentation . . . 214
6.140.3.1 Apply() . . . 214
6.140.4 Member Data Documentation . . . 215
6.140.4.1 Image . . . 215
6.140.5 Property Documentation . . . 215
6.140.5.1 ArcanaData . . . 215
6.140.5.2 ArcanaType . . . 215
6.141 CardHouse.TarotCardDefinition Class Reference . . . 215
6.141.1 Detailed Description . . . 216
6.141.2 Member Data Documentation . . . 216
6.141.2.1 Art . . . 216
6.141.2.2 Data . . . 216
6.142 CardHouse.SampleGames.Tarot.TarotSpread Class Reference . . . 216
6.142.1 Detailed Description . . . 216
6.142.2 Member Function Documentation . . . 216
6.142.2.1 FillNext() . . . 216
6.142.3 Member Data Documentation . . . 217
6.142.3.1 Instructions . . . 217
6.142.3.2 Name . . . 217
6.142.3.3 Slots . . . 217
6.143 CardHouse.TimedEvent Class Reference . . . 217
6.143.1 Detailed Description . . . 217
6.143.2 Member Function Documentation . . . 218
6.143.2.1 ActivateAndDelay() . . . 218
6.143.2.2 ExecuteChain() . . . 218
6.143.3 Member Data Documentation . . . 218
6.143.3.1 Duration . . . 218
6.143.3.2 Event . . . 218
6.144 CardHouse.Toggleable Class Reference . . . 218
6.144.1 Detailed Description . . . 219

6.144.2 Member Function Documentation . . . 219
6.144.2.1 SetIsActive() . . . 219
6.144.3 Member Data Documentation . . . 219
6.144.3.1 IsActive . . . 219
6.145 CardHouse.Tutorial.TransferOperatorTutorialUI Class Reference . . . 219
6.145.1 Detailed Description . . . 220
6.145.2 Member Function Documentation . . . 220
6.145.2.1 AdjustFlipSpeed() . . . 220
6.145.2.2 AdjustGrabFrom() . . . 220
6.145.2.3 AdjustNumberToTransfer() . . . 220
6.145.2.4 AdjustSendTo() . . . 220
6.145.3 Member Data Documentation . . . 220
6.145.3.1 FlipSpeedSlider . . . 220
6.145.3.2 FlipSpeedText . . . 221
6.145.3.3 GrabFromDropdown . . . 221
6.145.3.4 NumberToTransferSlider . . . 221
6.145.3.5 NumberToTransferText . . . 221
6.145.3.6 Operator . . . 221
6.145.3.7 SendToDropdown . . . 221
6.146 CardHouse.TriggerEnterRelay Class Reference . . . 222
6.146.1 Detailed Description . . . 222
6.146.2 Member Data Documentation . . . 222
6.146.2.1 Relay . . . 222
6.147 CardHouse.Turning Class Reference . . . 223
6.147.1 Detailed Description . . . 223
6.147.2 Member Function Documentation . . . 224
6.147.2.1 GetCurrentValue() . . . 224
6.147.2.2 GetDefaultSeeker() . . . 224
6.147.2.3 SetNewValue() . . . 224
6.148 CardHouse.Tutorial.TutorialButton Class Reference . . . 224
6.148.1 Detailed Description . . . 225
6.148.2 Member Function Documentation . . . 225
6.148.2.1 Setup() . . . 225
6.148.3 Member Data Documentation . . . 225
6.148.3.1 Label . . . 225
6.149 CardHouse.TweakVector3Seeker Class Reference . . . 225
6.149.1 Detailed Description . . . 226
6.149.2 Constructor & Destructor Documentation . . . 227
6.149.2.1 TweakVector3Seeker() . . . 227
6.149.3 Member Function Documentation . . . 227
6.149.3.1 MakeCopy() . . . 227
6.149.3.2 Pump() . . . 227

6.149.4 Member Data Documentation . . . 227
6.149.4.1 Tweak . . . 227
6.149.4.2 TweakMultiplier . . . 227
6.150 CardHouse.TweakVector3SeekerScriptable Class Reference . . . 228
6.150.1 Detailed Description . . . 228
6.150.2 Member Function Documentation . . . 228
6.150.2.1 GetStrategy() . . . 228
6.150.3 Member Data Documentation . . . 229
6.150.3.1 Tweak . . . 229
6.150.3.2 TweakMultiplier . . . 229
6.151 CardHouse.Tutorial.ValidDragTutorial Class Reference . . . 229
6.151.1 Detailed Description . . . 230
6.151.2 Member Function Documentation . . . 230
6.151.2.1 UpdateDropdown01() . . . 230
6.151.2.2 UpdateDropdown02() . . . 230
6.151.2.3 UpdateDropdown10() . . . 230
6.151.2.4 UpdateDropdown11() . . . 230
6.151.2.5 UpdateDropdown12() . . . 230
6.151.3 Member Data Documentation . . . 230
6.151.3.1 Dropdown01 . . . 230
6.151.3.2 Dropdown02 . . . 231
6.151.3.3 Dropdown10 . . . 231
6.151.3.4 Dropdown11 . . . 231
6.151.3.5 Dropdown12 . . . 231
6.151.3.6 GroupA . . . 231
6.151.3.7 GroupB . . . 231
6.151.3.8 GroupC . . . 231
6.151.3.9 GroupD . . . 231
6.151.3.10 PhaseManager . . . 232
6.152 CardHouse.Tutorial.WaterPlantAction Class Reference . . . 232
6.152.1 Detailed Description . . . 233
6.152.2 Member Function Documentation . . . 233
6.152.2.1 ActOnTarget() . . . 233
6.153 CardHouse.Tutorial.WaterTargetPlantGate Class Reference . . . 233
6.153.1 Detailed Description . . . 234
6.153.2 Member Function Documentation . . . 234
6.153.2.1 IsUnlockedInternal() . . . 234
6.154 CardHouse.WaypointCurveFloatAngleSeeker Class Reference . . . 234
6.154.1 Detailed Description . . . 235
6.154.2 Constructor & Destructor Documentation . . . 235
6.154.2.1 WaypointCurveFloatAngleSeeker() . . . 235
6.154.3 Member Function Documentation . . . 235

6.154.3.1 IsDone() . . . 235
6.154.3.2 MakeCopy() . . . 236
6.154.3.3 Pump() . . . 236
6.155 CardHouse.WaypointCurveFloatAngleSeekerScriptable Class Reference . . . 236
6.155.1 Detailed Description . . . 236
6.155.2 Member Function Documentation . . . 237
6.155.2.1 GetStrategy() . . . 237
6.155.3 Member Data Documentation . . . 237
6.155.3.1 Duration . . . 237
6.155.3.2 ProgressCurve . . . 237
6.156 CardHouse.WaypointCurveFloatSeeker Class Reference . . . 237
6.156.1 Detailed Description . . . 238
6.156.2 Constructor & Destructor Documentation . . . 238
6.156.2.1 WaypointCurveFloatSeeker() . . . 238
6.156.3 Member Function Documentation . . . 239
6.156.3.1 IsDone() . . . 239
6.156.3.2 MakeCopy() . . . 239
6.156.3.3 Pump() . . . 239
6.157 CardHouse.WaypointCurveFloatSeekerScriptable Class Reference . . . 239
6.157.1 Detailed Description . . . 240
6.157.2 Member Function Documentation . . . 240
6.157.2.1 GetStrategy() . . . 240
6.157.3 Member Data Documentation . . . 240
6.157.3.1 Duration . . . 240
6.157.3.2 ProgressCurve . . . 240
6.158 CardHouse.WaypointCurveVector3Seeker Class Reference . . . 240
6.158.1 Detailed Description . . . 241
6.158.2 Constructor & Destructor Documentation . . . 242
6.158.2.1 WaypointCurveVector3Seeker() . . . 242
6.158.3 Member Function Documentation . . . 242
6.158.3.1 IsDone() . . . 242
6.158.3.2 MakeCopy() . . . 242
6.158.3.3 Pump() . . . 242
6.159 CardHouse.WaypointCurveVector3SeekerScriptable Class Reference . . . 242
6.159.1 Detailed Description . . . 243
6.159.2 Member Function Documentation . . . 243
6.159.2.1 GetStrategy() . . . 243
6.159.3 Member Data Documentation . . . 243
6.159.3.1 Duration . . . 243
6.159.3.2 ProgressCurve . . . 243
6.160 CardHouse.TestScenes.WaypointTesterCard Class Reference . . . 244
6.160.1 Detailed Description . . . 244

6.160.2 Member Function Documentation . . . 244
6.160.2.1 Test() . . . 244
6.160.3 Member Data Documentation . . . 244
6.160.3.1 WaypointSeekers . . . 244
6.161 CardHouse.TestScenes.WaypointTesterGroup Class Reference . . . 245
6.161.1 Detailed Description . . . 245
6.161.2 Member Function Documentation . . . 245
6.161.2.1 Test() . . . 245
6.161.3 Member Data Documentation . . . 245
6.161.3.1 Waypoints . . . 245

7 File Documentation **247**

7.1 PokerCardDefinition.cs . . . 247
7.2 PokerCard.cs . . . 247
7.3 PokerSuit.cs . . . 247
7.4 ArcanaData.cs . . . 248
7.5 MajorArcanaName.cs . . . 248
7.6 TarotCard.cs . . . 248
7.7 TarotCardDefinition.cs . . . 249
7.8 TarotSuit.cs . . . 249
7.9 Card.cs . . . 249
7.10 CardFacing.cs . . . 251
7.11 Activatable.cs . . . 251
7.12 CardTargetCardOperator.cs . . . 252
7.13 CardTransferOperator.cs . . . 252
7.14 DiscardCardOperator.cs . . . 253
7.15 DiscardTargetCardOperator.cs . . . 253
7.16 ShuffleOperator.cs . . . 254
7.17 CardDefinition.cs . . . 254
7.18 DeckDefinition.cs . . . 254
7.19 ClickDetector.cs . . . 255
7.20 PhaseGateClick.cs . . . 255
7.21 GroupConditional.cs . . . 255
7.22 PhaseConditional.cs . . . 256
7.23 CurrencyChangeDetector.cs . . . 256
7.24 CurrencyContainer.cs . . . 257
7.25 CurrencyCost.cs . . . 257
7.26 CurrencyQuantity.cs . . . 257
7.27 CurrencyRegistry.cs . . . 258
7.28 CurrencyScriptable.cs . . . 259
7.29 CurrencyUI.cs . . . 260
7.30 CurrencyWallet.cs . . . 260

7.31 CurrencyOperator.cs . . . 260
7.32 CurrencyRefillOperator.cs . . . 261
7.33 IncrementCurrencyOperator.cs . . . 261
7.34 DragAction.cs . . . 261
7.35 DragDetector.cs . . . 262
7.36 Dragging.cs . . . 262
7.37 DragOperator.cs . . . 263
7.38 EventChain.cs . . . 264
7.39 TimedEvent.cs . . . 264
7.40 BlockAllDrops.cs . . . 265
7.41 CurrencyGate.cs . . . 265
7.42 Gate.cs . . . 265
7.43 GateCollection.cs . . . 266
7.44 DropParams.cs . . . 266
7.45 NoParams.cs . . . 266
7.46 TargetCardParams.cs . . . 266
7.47 CardGroup.cs . . . 267
7.48 GroupInteractability.cs . . . 273
7.49 GroupName.cs . . . 273
7.50 GroupRegistry.cs . . . 273
7.51 CardGridLayout.cs . . . 274
7.52 CardGroupSettings.cs . . . 275
7.53 SlotLayout.cs . . . 276
7.54 SplayLayout.cs . . . 276
7.55 StackLayout.cs . . . 277
7.56 MountDetector.cs . . . 277
7.57 MountingMode.cs . . . 278
7.58 DragTransition.cs . . . 278
7.59 GroupTransition.cs . . . 278
7.60 HoverDetector.cs . . . 278
7.61 TriggerEnterRelay.cs . . . 279
7.62 LifetimeDestructor.cs . . . 279
7.63 CardDropGateDimmer.cs . . . 279
7.64 CardLoyalty.cs . . . 281
7.65 DragGateDimmer.cs . . . 281
7.66 GroupDropGateDimmer.cs . . . 281
7.67 Loyalty.cs . . . 282
7.68 LoyaltyGateCardDrop.cs . . . 282
7.69 Phase.cs . . . 283
7.70 PhaseChangeDetector.cs . . . 283
7.71 PhaseGateCardDragStart.cs . . . 284
7.72 PhaseGateCardDrop.cs . . . 284

7.73 PhaseManager.cs . . . 284
7.74 BaseSeekerComponent.cs . . . 286
7.75 Homing.cs . . . 287
7.76 Scaling.cs . . . 287
7.77 Turning.cs . . . 288
7.78 AnimCurveFloatSeeker.cs . . . 288
7.79 AnimCurveFloatSeekerScriptable.cs . . . 289
7.80 ExponentialAngleFloatSeeker.cs . . . 289
7.81 ExponentialAngleFloatSeekerScriptable.cs . . . 289
7.82 ExponentialFloatSeeker.cs . . . 289
7.83 ExponentialFloatSeekerScriptable.cs . . . 290
7.84 InstantFloatSeeker.cs . . . 290
7.85 WaypointCurveFloatAngleSeeker.cs . . . 290
7.86 WaypointCurveFloatAngleSeekerScriptable.cs . . . 291
7.87 WaypointCurveFloatSeeker.cs . . . 291
7.88 WaypointCurveFloatSeekerScriptable.cs . . . 292
7.89 Seeker.cs . . . 292
7.90 SeekerScriptable.cs . . . 293
7.91 SeekerScriptableSet.cs . . . 293
7.92 SeekerSet.cs . . . 293
7.93 SeekerSetList.cs . . . 293
7.94 AnimCurveVector3Seeker.cs . . . 294
7.95 AnimCurveVector3SeekerScriptable.cs . . . 294
7.96 ContinuousInstantVector3Seeker.cs . . . 294
7.97 ContinuousInstantVector3SeekerScriptable.cs . . . 295
7.98 ExponentialVector3Seeker.cs . . . 295
7.99 ExponentialVector3SeekerScriptable.cs . . . 295
7.100 InstantVector3Seeker.cs . . . 296
7.101 RandomizedCurveVector3SeekerScriptable.cs . . . 296
7.102 TweakVector3Seeker.cs . . . 296
7.103 TweakVector3SeekerScriptable.cs . . . 297
7.104 WaypointCurveVector3Seeker.cs . . . 297
7.105 WaypointCurveVector3SeekerScriptable.cs . . . 297
7.106 CardSetup.cs . . . 298
7.107 DeckSetup.cs . . . 298
7.108 GroupSetup.cs . . . 299
7.109 MultiplayerBoardSetup.cs . . . 300
7.110 MultiSpriteOperator.cs . . . 305
7.111 SpriteColorOperator.cs . . . 306
7.112 SpriteImageOperator.cs . . . 306
7.113 SpriteOperator.cs . . . 307
7.114 Toggleable.cs . . . 308

7.115 Utils.cs . . . 308
7.116 DamageGroupOperator.cs . . . 309
7.117 DamageTargetOperator.cs . . . 309
7.118 Health.cs . . . 309
7.119 MemoryCard.cs . . . 310
7.120 MemoryGame.cs . . . 310
7.121 MemoryUI.cs . . . 312
7.122 SolitaireCardDragHandler.cs . . . 312
7.123 SolitaireColumnChangeHandler.cs . . . 313
7.124 SolitaireColumnDropGate.cs . . . 313
7.125 SolitaireDeckClickHandler.cs . . . 313
7.126 SolitaireScorePileDropGate.cs . . . 314
7.127 SolitaireSetup.cs . . . 314
7.128 SpreadManager.cs . . . 315
7.129 TarotSpread.cs . . . 316
7.130 WaypointTesterCard.cs . . . 317
7.131 WaypointTesterGroup.cs . . . 317
7.132 LaunchTutorialOption.cs . . . 318
7.133 LaunchDataScriptable.cs . . . 319
7.134 CardDragTutorial.cs . . . 320
7.135 GroupSetupTutorial.cs . . . 321
7.136 ClosestCardHighlighter.cs . . . 322
7.137 StackTutorial.cs . . . 322
7.138 SplayTutorial.cs . . . 323
7.139 GridTutorial.cs . . . 324
7.140 DiscardAllCardsOperator.cs . . . 325
7.141 SpriteOperatorTutorial.cs . . . 325
7.142 SpriteVoterTutorial.cs . . . 326
7.143 TransferOperatorTutorialUI.cs . . . 327
7.144 SeekerTutorial.cs . . . 328
7.145 EventChainsTutorial.cs . . . 329
7.146 ValidDragTutorial.cs . . . 329
7.147 MatureCropDragGate.cs . . . 330
7.148 PhaseLabelUpdater.cs . . . 330
7.149 Plant.cs . . . 331
7.150 PlantGrowthScriptable.cs . . . 331
7.151 WaterPlantAction.cs . . . 332
7.152 WaterTargetPlantGate.cs . . . 332
7.153 PresentationPointTutorial.cs . . . 332
7.154 MultiBoardTutorial.cs . . . 333
7.155 OutLinks.cs . . . 335
7.156 SandboxManager.cs . . . 335

7.157 SceneKeeper.cs 336
7.158 SceneSpawner.cs 336
7.159 TutorialButton.cs 337
7.160 StringListScriptable.cs 337

Index **339**

Chapter 1

Namespace Index

1.1 Package List

Here are the packages with brief descriptions (if available):

CardHouse 15
CardHouse.SampleGames 19
CardHouse.SampleGames.DeckBuilder 19
CardHouse.SampleGames.MemoryMatch 19
CardHouse.SampleGames.Solitaire 19
CardHouse.SampleGames.Tarot 19
CardHouse.TestScenes 20
CardHouse.Tutorial 20

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

CardHouse.ArcanaData . . . 29
CardHouse.BaseSeekerComponent< float > . . . 30
  CardHouse.Scaling . . . 174
  CardHouse.Turning . . . 223
CardHouse.BaseSeekerComponent< Vector3 > . . . 30
  CardHouse.Homing . . . 123
CardHouse.CurrencyCost.CostWithLabel . . . 60
CardHouse.DropParams . . . 92
CardHouse.Gate< DropParams > . . . 104
  CardHouse.BlockAllDrops . . . 31
  CardHouse.LoyaltyGateCardDrop . . . 132
  CardHouse.PhaseGateCardDrop . . . 155
  CardHouse.SampleGames.Solitaire.SolitaireColumnDropGate . . . 187
  CardHouse.SampleGames.Solitaire.SolitaireScorePileDropGate . . . 189
CardHouse.Gate< NoParams > . . . 104
  CardHouse.CurrencyGate . . . 65
  CardHouse.PhaseGateCardDragStart . . . 154
  CardHouse.PhaseGateClick . . . 156
  CardHouse.Tutorial.MatureCropDragGate . . . 133
CardHouse.Gate< TargetCardParams > . . . 104
  CardHouse.Tutorial.WaterTargetPlantGate . . . 233
CardHouse.GateCollection< T > . . . 105
CardHouse.GateCollection< CardHouse.DropParams > . . . 105
CardHouse.GateCollection< CardHouse.NoParams > . . . 105
CardHouse.GateCollection< CardHouse.TargetCardParams > . . . 105
CardHouse.GroupNameUnityActionKvp . . . 111
CardHouse.GroupSetup.GroupPopulationData . . . 112
CardHouse.GroupTransition . . . 119
  CardHouse.DragTransition . . . 91
CardHouse.MultiplayerBoardSetup.GroupTransitionByName . . . 120
CardHouse.Card.GroupTransitionEvent . . . 121
ICloneable
  CardHouse.CurrencyQuantity . . . 67

CardHouse.CurrencyContainer . . . 62
CardHouse.CurrencyWallet . . . 73
CardHouse.Tutorial.MultiBoardTutorial.InstructionImagePair . . . 129
IPointerClickHandler
CardHouse.Tutorial.OutLinks . . . 149
List
CardHouse.SeekerSetList . . . 180
MonoBehaviour
CardHouse.Activatable . . . 21
CardHouse.CardTargetCardOperator . . . 53
CardHouse.DiscardTargetCardOperator . . . 82
CardHouse.SampleGames.DeckBuilder.DamageGroupOperator . . . 75
CardHouse.SampleGames.DeckBuilder.DamageTargetOperator . . . 76
CardHouse.Tutorial.WaterPlantAction . . . 232
CardHouse.CardTransferOperator . . . 54
CardHouse.GroupConditional . . . 109
CardHouse.PhaseConditional . . . 153
CardHouse.ShuffleOperator . . . 183
CardHouse.BaseSeekerComponent< T > . . . 30
CardHouse.Card . . . 32
CardHouse.CardGroup . . . 44
CardHouse.CardGroupSettings . . . 50
CardHouse.CardGridLayout . . . 42
CardHouse.SlotLayout . . . 184
CardHouse.SplayLayout . . . 192
CardHouse.StackLayout . . . 207
CardHouse.CardLoyalty . . . 51
CardHouse.CardSetup . . . 52
CardHouse.PokerCard . . . 164
CardHouse.TarotCard . . . 213
CardHouse.CurrencyChangeDetector . . . 61
CardHouse.CurrencyCost . . . 64
CardHouse.CurrencyOperator . . . 66
CardHouse.CurrencyRefillOperator . . . 68
CardHouse.IncrementCurrencyOperator . . . 125
CardHouse.CurrencyRegistry . . . 69
CardHouse.CurrencyUI . . . 72
CardHouse.DeckSetup . . . 79
CardHouse.DiscardCardOperator . . . 81
CardHouse.DragOperator . . . 89
CardHouse.Dragging . . . 86
CardHouse.EventChain . . . 93
CardHouse.GroupDropGateDimmer . . . 110
CardHouse.GroupRegistry . . . 113
CardHouse.GroupSetup . . . 115
CardHouse.LifetimeDestructor . . . 131
CardHouse.MountDetector . . . 139
CardHouse.MultiSpriteOperator . . . 145
CardHouse.MultiplayerBoardSetup . . . 143
CardHouse.PhaseChangeDetector . . . 152
CardHouse.PhaseManager . . . 158
CardHouse.SampleGames.DeckBuilder.Health . . . 122
CardHouse.SampleGames.MemoryMatch.MemoryCard . . . 134
CardHouse.SampleGames.MemoryMatch.MemoryGame . . . 135
CardHouse.SampleGames.MemoryMatch.MemoryUI . . . 137
CardHouse.SampleGames.Solitaire.SolitaireCardDragHandler . . . 185
CardHouse.SampleGames.Solitaire.SolitaireColumnChangeHandler . . . 186
CardHouse.SampleGames.Solitaire.SolitaireDeckClickHandler . . . 188

CardHouse.SampleGames.Solitaire.SolitaireSetup . . . 190
CardHouse.SampleGames.Tarot.SpreadManager . . . 197
CardHouse.SpriteOperator . . . 202
CardHouse.SpriteColorOperator . . . 199
CardHouse.SpriteImageOperator . . . 201
CardHouse.TestScenes.WaypointTesterCard . . . 244
CardHouse.TestScenes.WaypointTesterGroup . . . 245
CardHouse.Toggleable . . . 218
CardHouse.CardDropGateDimmer . . . 41
CardHouse.ClickDetector . . . 56
CardHouse.DragDetector . . . 83
CardHouse.DragGateDimmer . . . 85
CardHouse.Gate< T > . . . 104
CardHouse.HoverDetector . . . 124
CardHouse.TriggerEnterRelay . . . 222
CardHouse.Tutorial.CardDragTutorial . . . 38
CardHouse.Tutorial.ClosestCardHighlighter . . . 58
CardHouse.Tutorial.DiscardAllCardsOperator . . . 80
CardHouse.Tutorial.EventChainsTutorial . . . 94
CardHouse.Tutorial.GridTutorial . . . 106
CardHouse.Tutorial.GroupSetupTutorial . . . 116
CardHouse.Tutorial.MultiBoardTutorial . . . 139
CardHouse.Tutorial.OutLinks . . . 149
CardHouse.Tutorial.PhaseLabelUpdater . . . 157
CardHouse.Tutorial.Plant . . . 161
CardHouse.Tutorial.PresentationPointTutorial . . . 167
CardHouse.Tutorial.SandboxManager . . . 171
CardHouse.Tutorial.SceneKeeper . . . 176
CardHouse.Tutorial.SceneSpawner . . . 176
CardHouse.Tutorial.SeekerTutorial . . . 181
CardHouse.Tutorial.SplayTutorial . . . 194
CardHouse.Tutorial.SpriteOperatorTutorial . . . 204
CardHouse.Tutorial.SpriteVoterTutorial . . . 206
CardHouse.Tutorial.StackTutorial . . . 209
CardHouse.Tutorial.TransferOperatorTutorialUI . . . 219
CardHouse.Tutorial.TutorialButton . . . 224
CardHouse.Tutorial.ValidDragTutorial . . . 229
CardHouse.SpriteColorOperator.NamedColor . . . 146
CardHouse.GroupRegistry.NamedGroup . . . 147
CardHouse.SpriteImageOperator.NamedSprite . . . 148
CardHouse.NoParams . . . 148
CardHouse.Phase . . . 150
CardHouse.Tutorial.PlantMaturityInfo . . . 164
ScriptableObject
CardHouse.CardDefinition . . . 37
CardHouse.PokerCardDefinition . . . 166
CardHouse.TarotCardDefinition . . . 215
CardHouse.CurrencyScriptable . . . 72
CardHouse.DeckDefinition . . . 78
CardHouse.SeekerScriptable< T > . . . 178
CardHouse.Tutorial.LaunchDataScriptable . . . 130
CardHouse.Tutorial.PlantGrowthScriptable . . . 163
CardHouse.Tutorial.StringListScriptable . . . 211
CardHouse.Seeker< T > . . . 177
CardHouse.Seeker< float > . . . 177
CardHouse.AnimCurveFloatSeeker . . . 22
CardHouse.WaypointCurveFloatAngleSeeker . . . 234
CardHouse.WaypointCurveFloatSeeker . . . 237

CardHouse.ExponentialFloatSeeker . . . 98
CardHouse.ExponentialAngleFloatSeeker . . . 95
CardHouse.InstantFloatSeeker . . . 127
CardHouse.Seeker< Vector3 > . . . 177
CardHouse.AnimCurveVector3Seeker . . . 25
CardHouse.TweakVector3Seeker . . . 225
CardHouse.WaypointCurveVector3Seeker . . . 240
CardHouse.ContinuousInstantVector3Seeker . . . 58
CardHouse.ExponentialVector3Seeker . . . 101
CardHouse.InstantVector3Seeker . . . 128
CardHouse.SeekerScriptable< float > . . . 178
CardHouse.AnimCurveFloatSeekerScriptable . . . 24
CardHouse.ExponentialAngleFloatSeekerScriptable . . . 97
CardHouse.ExponentialFloatSeekerScriptable . . . 100
CardHouse.WaypointCurveFloatAngleSeekerScriptable . . . 236
CardHouse.WaypointCurveFloatSeekerScriptable . . . 239
CardHouse.SeekerScriptable< Vector3 > . . . 178
CardHouse.AnimCurveVector3SeekerScriptable . . . 28
CardHouse.TweakVector3SeekerScriptable . . . 228
CardHouse.RandomizedCurveVector3SeekerScriptable . . . 170
CardHouse.ContinuousInstantVector3SeekerScriptable . . . 59
CardHouse.ExponentialVector3SeekerScriptable . . . 103
CardHouse.WaypointCurveVector3SeekerScriptable . . . 242
CardHouse.SeekerScriptableSet . . . 178
CardHouse.SeekerSet . . . 179
CardHouse.Tutorial.StringSeekerKVP . . . 212
CardHouse.StringUnityActionKvp . . . 212
CardHouse.TargetCardParams . . . 213
CardHouse.SampleGames.Tarot.TarotSpread . . . 216
CardHouse.TimedEvent . . . 217

Chapter 3

Class Index

3.1 Class List

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

CardHouse.Activatable . . . 21
CardHouse.AnimCurveFloatSeeker . . . 22
CardHouse.AnimCurveFloatSeekerScriptable . . . 24
CardHouse.AnimCurveVector3Seeker . . . 25
CardHouse.AnimCurveVector3SeekerScriptable . . . 28
CardHouse.ArcanaData . . . 29
CardHouse.BaseSeekerComponent< T > . . . 30
CardHouse.BlockAllDrops . . . 31
CardHouse.Card . . . 32
CardHouse.CardDefinition . . . 37
CardHouse.Tutorial.CardDragTutorial . . . 38
CardHouse.CardDropGateDimmer . . . 41
CardHouse.CardGridLayout . . . 42
CardHouse.CardGroup . . . 44
CardHouse.CardGroupSettings . . . 50
CardHouse.CardLoyalty . . . 51
CardHouse.CardSetup . . . 52
CardHouse.CardTargetCardOperator . . . 53
CardHouse.CardTransferOperator . . . 54
CardHouse.ClickDetector . . . 56
CardHouse.Tutorial.ClosestCardHighlighter . . . 58
CardHouse.ContinuousInstantVector3Seeker . . . 58
CardHouse.ContinuousInstantVector3SeekerScriptable . . . 59
CardHouse.CurrencyCost.CostWithLabel . . . 60
CardHouse.CurrencyChangeDetector . . . 61
CardHouse.CurrencyContainer . . . 62
CardHouse.CurrencyCost . . . 64
CardHouse.CurrencyGate . . . 65
CardHouse.CurrencyOperator . . . 66
CardHouse.CurrencyQuantity . . . 67
CardHouse.CurrencyRefillOperator . . . 68
CardHouse.CurrencyRegistry . . . 69
CardHouse.CurrencyScriptable . . . 72
CardHouse.CurrencyUI . . . 72
CardHouse.CurrencyWallet . . . 73

CardHouse.SampleGames.DeckBuilder.DamageGroupOperator . . . 75
CardHouse.SampleGames.DeckBuilder.DamageTargetOperator . . . 76
CardHouse.DeckDefinition . . . 78
CardHouse.DeckSetup . . . 79
CardHouse.Tutorial.DiscardAllCardsOperator . . . 80
CardHouse.DiscardCardOperator . . . 81
CardHouse.DiscardTargetCardOperator . . . 82
CardHouse.DragDetector . . . 83
CardHouse.DragGateDimmer . . . 85
CardHouse.Dragging . . . 86
CardHouse.DragOperator . . . 89
CardHouse.DragTransition . . . 91
CardHouse.DropParams . . . 92
CardHouse.EventChain . . . 93
CardHouse.Tutorial.EventChainsTutorial . . . 94
CardHouse.ExponentialAngleFloatSeeker . . . 95
CardHouse.ExponentialAngleFloatSeekerScriptable . . . 97
CardHouse.ExponentialFloatSeeker . . . 98
CardHouse.ExponentialFloatSeekerScriptable . . . 100
CardHouse.ExponentialVector3Seeker . . . 101
CardHouse.ExponentialVector3SeekerScriptable . . . 103
CardHouse.Gate< T > . . . 104
CardHouse.GateCollection< T > . . . 105
CardHouse.Tutorial.GridTutorial . . . 106
CardHouse.GroupConditional . . . 109
CardHouse.GroupDropGateDimmer . . . 110
CardHouse.GroupNameUnityActionKvp . . . 111
CardHouse.GroupSetup.GroupPopulationData . . . 112
CardHouse.GroupRegistry . . . 113
CardHouse.GroupSetup . . . 115
CardHouse.Tutorial.GroupSetupTutorial . . . 116
CardHouse.GroupTransition . . . 119
CardHouse.MultiplayerBoardSetup.GroupTransitionByName . . . 120
CardHouse.Card.GroupTransitionEvent . . . 121
CardHouse.SampleGames.DeckBuilder.Health . . . 122
CardHouse.Homing . . . 123
CardHouse.HoverDetector . . . 124
CardHouse.IncrementCurrencyOperator . . . 125
CardHouse.InstantFloatSeeker . . . 127
CardHouse.InstantVector3Seeker . . . 128
CardHouse.Tutorial.MultiBoardTutorial.InstructionImagePair . . . 129
CardHouse.Tutorial.LaunchDataScriptable . . . 130
CardHouse.LifetimeDestructor . . . 131
CardHouse.LoyaltyGateCardDrop . . . 132
CardHouse.Tutorial.MatureCropDragGate . . . 133
CardHouse.SampleGames.MemoryMatch.MemoryCard . . . 134
CardHouse.SampleGames.MemoryMatch.MemoryGame . . . 135
CardHouse.SampleGames.MemoryMatch.MemoryUI . . . 137
CardHouse.MountDetector . . . 139
CardHouse.Tutorial.MultiBoardTutorial . . . 139
CardHouse.MultiplayerBoardSetup . . . 143
CardHouse.MultiSpriteOperator . . . 145
CardHouse.SpriteColorOperator.NamedColor . . . 146
CardHouse.GroupRegistry.NamedGroup . . . 147
CardHouse.SpriteImageOperator.NamedSprite . . . 148
CardHouse.NoParams . . . 148
CardHouse.Tutorial.OutLinks . . . 149
CardHouse.Phase . . . 150

CardHouse.PhaseChangeDetector . . . 152
CardHouse.PhaseConditional . . . 153
CardHouse.PhaseGateCardDragStart . . . 154
CardHouse.PhaseGateCardDrop . . . 155
CardHouse.PhaseGateClick . . . 156
CardHouse.Tutorial.PhaseLabelUpdater . . . 157
CardHouse.PhaseManager . . . 158
CardHouse.Tutorial.Plant . . . 161
CardHouse.Tutorial.PlantGrowthScriptable . . . 163
CardHouse.Tutorial.PlantMaturityInfo . . . 164
CardHouse.PokerCard . . . 164
CardHouse.PokerCardDefinition . . . 166
CardHouse.Tutorial.PresentationPointTutorial . . . 167
CardHouse.RandomizedCurveVector3SeekerScriptable . . . 170
CardHouse.Tutorial.SandboxManager . . . 171
CardHouse.Scaling . . . 174
CardHouse.Tutorial.SceneKeeper . . . 176
CardHouse.Tutorial.SceneSpawner . . . 176
CardHouse.Seeker< T > . . . 177
CardHouse.SeekerScriptable< T > . . . 178
CardHouse.SeekerScriptableSet . . . 178
CardHouse.SeekerSet . . . 179
CardHouse.SeekerSetList . . . 180
CardHouse.Tutorial.SeekerTutorial . . . 181
CardHouse.ShuffleOperator . . . 183
CardHouse.SlotLayout . . . 184
CardHouse.SampleGames.Solitaire.SolitaireCardDragHandler . . . 185
CardHouse.SampleGames.Solitaire.SolitaireColumnChangeHandler . . . 186
CardHouse.SampleGames.Solitaire.SolitaireColumnDropGate . . . 187
CardHouse.SampleGames.Solitaire.SolitaireDeckClickHandler . . . 188
CardHouse.SampleGames.Solitaire.SolitaireScorePileDropGate . . . 189
CardHouse.SampleGames.Solitaire.SolitaireSetup . . . 190
CardHouse.SplayLayout . . . 192
CardHouse.Tutorial.SplayTutorial . . . 194
CardHouse.SampleGames.Tarot.SpreadManager . . . 197
CardHouse.SpriteColorOperator . . . 199
CardHouse.SpriteImageOperator . . . 201
CardHouse.SpriteOperator . . . 202
CardHouse.Tutorial.SpriteOperatorTutorial . . . 204
CardHouse.Tutorial.SpriteVoterTutorial . . . 206
CardHouse.StackLayout . . . 207
CardHouse.Tutorial.StackTutorial . . . 209
CardHouse.Tutorial.StringListScriptable . . . 211
CardHouse.Tutorial.StringSeekerKVP . . . 212
CardHouse.StringUnityActionKvp . . . 212
CardHouse.TargetCardParams . . . 213
CardHouse.TarotCard . . . 213
CardHouse.TarotCardDefinition . . . 215
CardHouse.SampleGames.Tarot.TarotSpread . . . 216
CardHouse.TimedEvent . . . 217
CardHouse.Toggleable . . . 218
CardHouse.Tutorial.TransferOperatorTutorialUI . . . 219
CardHouse.TriggerEnterRelay . . . 222
CardHouse.Turning . . . 223
CardHouse.Tutorial.TutorialButton . . . 224
CardHouse.TweakVector3Seeker . . . 225
CardHouse.TweakVector3SeekerScriptable . . . 228
CardHouse.Tutorial.ValidDragTutorial . . . 229

CardHouse.Tutorial.WaterPlantAction 232
CardHouse.Tutorial.WaterTargetPlantGate 233
CardHouse.WaypointCurveFloatAngleSeeker 234
CardHouse.WaypointCurveFloatAngleSeekerScriptable 236
CardHouse.WaypointCurveFloatSeeker 237
CardHouse.WaypointCurveFloatSeekerScriptable 239
CardHouse.WaypointCurveVector3Seeker 240
CardHouse.WaypointCurveVector3SeekerScriptable 242
CardHouse.TestScenes.WaypointTesterCard 244
CardHouse.TestScenes.WaypointTesterGroup 245

Chapter 4

File Index

4.1 File List

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

PokerCardDefinition.cs . . . 247
PokerCard.cs . . . 247
PokerSuit.cs . . . 247
ArcanaData.cs . . . 248
MajorArcanaName.cs . . . 248
TarotCard.cs . . . 248
TarotCardDefinition.cs . . . 249
TarotSuit.cs . . . 249
Card.cs . . . 249
CardFacing.cs . . . 251
Activatable.cs . . . 251
CardTargetCardOperator.cs . . . 252
CardTransferOperator.cs . . . 252
DiscardCardOperator.cs . . . 253
DiscardTargetCardOperator.cs . . . 253
ShuffleOperator.cs . . . 254
CardDefinition.cs . . . 254
DeckDefinition.cs . . . 254
ClickDetector.cs . . . 255
PhaseGateClick.cs . . . 255
GroupConditional.cs . . . 255
PhaseConditional.cs . . . 256
CurrencyChangeDetector.cs . . . 256
CurrencyContainer.cs . . . 257
CurrencyCost.cs . . . 257
CurrencyQuantity.cs . . . 257
CurrencyRegistry.cs . . . 258
CurrencyScriptable.cs . . . 259
CurrencyUI.cs . . . 260
CurrencyWallet.cs . . . 260
CurrencyOperator.cs . . . 260
CurrencyRefillOperator.cs . . . 261
IncrementCurrencyOperator.cs . . . 261
DragAction.cs . . . 261
DragDetector.cs . . . 262

Dragging.cs . . . 262
DragOperator.cs . . . 263
EventChain.cs . . . 264
TimedEvent.cs . . . 264
BlockAllDrops.cs . . . 265
CurrencyGate.cs . . . 265
Gate.cs . . . 265
GateCollection.cs . . . 266
DropParams.cs . . . 266
NoParams.cs . . . 266
TargetCardParams.cs . . . 266
CardGroup.cs . . . 267
GroupInteractability.cs . . . 273
GroupName.cs . . . 273
GroupRegistry.cs . . . 273
CardGridLayout.cs . . . 274
CardGroupSettings.cs . . . 275
SlotLayout.cs . . . 276
SplayLayout.cs . . . 276
StackLayout.cs . . . 277
MountDetector.cs . . . 277
MountingMode.cs . . . 278
DragTransition.cs . . . 278
GroupTransition.cs . . . 278
HoverDetector.cs . . . 278
TriggerEnterRelay.cs . . . 279
LifetimeDestructor.cs . . . 279
CardDropGateDimmer.cs . . . 279
CardLoyalty.cs . . . 281
DragGateDimmer.cs . . . 281
GroupDropGateDimmer.cs . . . 281
Loyalty.cs . . . 282
LoyaltyGateCardDrop.cs . . . 282
Phase.cs . . . 283
PhaseChangeDetector.cs . . . 283
PhaseGateCardDragStart.cs . . . 284
PhaseGateCardDrop.cs . . . 284
PhaseManager.cs . . . 284
BaseSeekerComponent.cs . . . 286
Homing.cs . . . 287
Scaling.cs . . . 287
Turning.cs . . . 288
AnimCurveFloatSeeker.cs . . . 288
AnimCurveFloatSeekerScriptable.cs . . . 289
ExponentialAngleFloatSeeker.cs . . . 289
ExponentialAngleFloatSeekerScriptable.cs . . . 289
ExponentialFloatSeeker.cs . . . 289
ExponentialFloatSeekerScriptable.cs . . . 290
InstantFloatSeeker.cs . . . 290
WaypointCurveFloatAngleSeeker.cs . . . 290
WaypointCurveFloatAngleSeekerScriptable.cs . . . 291
WaypointCurveFloatSeeker.cs . . . 291
WaypointCurveFloatSeekerScriptable.cs . . . 292
Seeker.cs . . . 292
SeekerScriptable.cs . . . 293
SeekerScriptableSet.cs . . . 293
SeekerSet.cs . . . 293
SeekerSetList.cs . . . 293

AnimCurveVector3Seeker.cs . . . 294
AnimCurveVector3SeekerScriptable.cs . . . 294
ContinuousInstantVector3Seeker.cs . . . 294
ContinuousInstantVector3SeekerScriptable.cs . . . 295
ExponentialVector3Seeker.cs . . . 295
ExponentialVector3SeekerScriptable.cs . . . 295
InstantVector3Seeker.cs . . . 296
RandomizedCurveVector3SeekerScriptable.cs . . . 296
TweakVector3Seeker.cs . . . 296
TweakVector3SeekerScriptable.cs . . . 297
WaypointCurveVector3Seeker.cs . . . 297
WaypointCurveVector3SeekerScriptable.cs . . . 297
CardSetup.cs . . . 298
DeckSetup.cs . . . 298
GroupSetup.cs . . . 299
MultiplayerBoardSetup.cs . . . 300
MultiSpriteOperator.cs . . . 305
SpriteColorOperator.cs . . . 306
SpriteImageOperator.cs . . . 306
SpriteOperator.cs . . . 307
Toggleable.cs . . . 308
Utils.cs . . . 308
DamageGroupOperator.cs . . . 309
DamageTargetOperator.cs . . . 309
Health.cs . . . 309
MemoryCard.cs . . . 310
MemoryGame.cs . . . 310
MemoryUI.cs . . . 312
SolitaireCardDragHandler.cs . . . 312
SolitaireColumnChangeHandler.cs . . . 313
SolitaireColumnDropGate.cs . . . 313
SolitaireDeckClickHandler.cs . . . 313
SolitaireScorePileDropGate.cs . . . 314
SolitaireSetup.cs . . . 314
SpreadManager.cs . . . 315
TarotSpread.cs . . . 316
WaypointTesterCard.cs . . . 317
WaypointTesterGroup.cs . . . 317
LaunchTutorialOption.cs . . . 318
LaunchDataScriptable.cs . . . 319
CardDragTutorial.cs . . . 320
GroupSetupTutorial.cs . . . 321
ClosestCardHighlighter.cs . . . 322
StackTutorial.cs . . . 322
SplayTutorial.cs . . . 323
GridTutorial.cs . . . 324
DiscardAllCardsOperator.cs . . . 325
SpriteOperatorTutorial.cs . . . 325
SpriteVoterTutorial.cs . . . 326
TransferOperatorTutorialUI.cs . . . 327
SeekerTutorial.cs . . . 328
EventChainsTutorial.cs . . . 329
ValidDragTutorial.cs . . . 329
MatureCropDragGate.cs . . . 330
PhaseLabelUpdater.cs . . . 330
Plant.cs . . . 331
PlantGrowthScriptable.cs . . . 331
WaterPlantAction.cs . . . 332

WaterTargetPlantGate.cs 332
PresentationPointTutorial.cs 332
MultiBoardTutorial.cs 333
OutLinks.cs 335
SandboxManager.cs 335
SceneKeeper.cs 336
SceneSpawner.cs 336
TutorialButton.cs 337
StringListScriptable.cs 337

Chapter 5

Namespace Documentation

5.1 CardHouse Namespace Reference

Classes

- class Activatable
- class AnimCurveFloatSeeker
- class AnimCurveFloatSeekerScriptable
- class AnimCurveVector3Seeker
- class AnimCurveVector3SeekerScriptable
- class ArcanaData
- class BaseSeekerComponent
- class BlockAllDrops
- class Card
- class CardDefinition
- class CardDropGateDimmer
- class CardGridLayout
- class CardGroup
- class CardGroupSettings
- class CardLoyalty
- class CardSetup
- class CardTargetCardOperator
- class CardTransferOperator
- class ClickDetector
- class ContinuousInstantVector3Seeker
- class ContinuousInstantVector3SeekerScriptable
- class CurrencyChangeDetector
- class CurrencyContainer
- class CurrencyCost
- class CurrencyGate
- class CurrencyOperator
- class CurrencyQuantity
- class CurrencyRefillOperator
- class CurrencyRegistry
- class CurrencyScriptable
- class CurrencyUI
- class CurrencyWallet
- class DeckDefinition

- class DeckSetup
- class DiscardCardOperator
- class DiscardTargetCardOperator
- class DragDetector
- class DragGateDimmer
- class Dragging
- class DragOperator
- class DragTransition
- class DropParams
- class EventChain
- class ExponentialAngleFloatSeeker
- class ExponentialAngleFloatSeekerScriptable
- class ExponentialFloatSeeker
- class ExponentialFloatSeekerScriptable
- class ExponentialVector3Seeker
- class ExponentialVector3SeekerScriptable
- class Gate
- class GateCollection
- class GroupConditional
- class GroupDropGateDimmer
- class GroupNameUnityActionKvp
- class GroupRegistry
- class GroupSetup
- class GroupTransition
- class Homing
- class HoverDetector
- class IncrementCurrencyOperator
- class InstantFloatSeeker
- class InstantVector3Seeker
- class LifetimeDestructor
- class LoyaltyGateCardDrop
- class MountDetector
- class MultiplayerBoardSetup
- class MultiSpriteOperator
- class NoParams
- class Phase
- class PhaseChangeDetector
- class PhaseConditional
- class PhaseGateCardDragStart
- class PhaseGateCardDrop
- class PhaseGateClick
- class PhaseManager
- class PokerCard
- class PokerCardDefinition
- class RandomizedCurveVector3SeekerScriptable
- class Scaling
- class Seeker
- class SeekerScriptable
- class SeekerScriptableSet
- class SeekerSet
- class SeekerSetList
- class ShuffleOperator
- class SlotLayout
- class SplayLayout
- class SpriteColorOperator

- class SpriteImageOperator
- class SpriteOperator
- class StackLayout
- class StringUnityActionKvp
- class TargetCardParams
- class TarotCard
- class TarotCardDefinition
- class TimedEvent
- class Toggleable
- class TriggerEnterRelay
- class Turning
- class TweakVector3Seeker
- class TweakVector3SeekerScriptable
- class **Utils**
- class WaypointCurveFloatAngleSeeker
- class WaypointCurveFloatAngleSeekerScriptable
- class WaypointCurveFloatSeeker
- class WaypointCurveFloatSeekerScriptable
- class WaypointCurveVector3Seeker
- class WaypointCurveVector3SeekerScriptable

Enumerations

- enum **PokerSuit** {
  None , **Hearts** , **Spades** , **Clubs** ,
  Diamonds }
- enum **MajorArcanaName** {
  None , **Fool** , **Magician** , **HighPriestess** ,
  Empress , **Emperor** , **Hierophant** , **Lovers** ,
  Chariot , **Strength** , **Hermit** , **WheelOfFortune** ,
  Justice , **HangedMan** , **Death** , **Temperance** ,
  Devil , **Tower** , **Star** , **Moon** ,
  Judgement , **World** , **Sun** }
- enum **TarotSuit** {
  None , **Swords** , **Cups** , **Pentacles** ,
  Wands }
- enum **CardFacing** { **None** , **FaceUp** , **FaceDown** }
- enum **DragAction** { **None** , **Mount** , **UseAndDiscard** , **UseOnTargetAndDiscard** }
- enum **GroupTargetType** { **First** , **Last** , **Random** }
- enum **GroupInteractability** { **None** , **Active** , **Inactive** , **OnlyTopActive** }
- enum **GroupName** {
  None , **Discard** , **Deck** , **Hand** ,
  Board , **A** , **B** , **C** ,
  D }
- enum **MountingMode** { **Top** , **Bottom** , **Closest** }
- enum **Loyalty** { **None** = 0 , **Self** = 1 , **Other** = 2 }

5.1.1 Enumeration Type Documentation

5.1.1.1 CardFacing

`enum CardHouse.CardFacing`

Definition at line 3 of file CardFacing.cs.

5.1.1.2 DragAction

`enum CardHouse.DragAction`

Definition at line 3 of file DragAction.cs.

5.1.1.3 GroupInteractability

`enum CardHouse.GroupInteractability`

Definition at line 3 of file GroupInteractability.cs.

5.1.1.4 GroupName

`enum CardHouse.GroupName`

Definition at line 3 of file GroupName.cs.

5.1.1.5 GroupTargetType

`enum CardHouse.GroupTargetType`

Definition at line 517 of file CardGroup.cs.

5.1.1.6 Loyalty

`enum CardHouse.Loyalty`

Definition at line 6 of file Loyalty.cs.

5.1.1.7 MajorArcanaName

`enum CardHouse.MajorArcanaName`

Definition at line 3 of file MajorArcanaName.cs.

5.1.1.8 MountingMode

`enum CardHouse.MountingMode`

Definition at line 3 of file MountingMode.cs.

5.1.1.9 PokerSuit

`enum CardHouse.PokerSuit`

Definition at line 3 of file PokerSuit.cs.

5.1.1.10 TarotSuit

```
enum CardHouse.TarotSuit
```

Definition at line 3 of file TarotSuit.cs.

5.2 CardHouse.SampleGames Namespace Reference

5.3 CardHouse.SampleGames.DeckBuilder Namespace Reference

Classes

- class DamageGroupOperator
- class DamageTargetOperator
- class Health

5.4 CardHouse.SampleGames.MemoryMatch Namespace Reference

Classes

- class MemoryCard
- class MemoryGame
- class MemoryUI

5.5 CardHouse.SampleGames.Solitaire Namespace Reference

Classes

- class SolitaireCardDragHandler
- class SolitaireColumnChangeHandler
- class SolitaireColumnDropGate
- class SolitaireDeckClickHandler
- class SolitaireScorePileDropGate
- class SolitaireSetup

5.6 CardHouse.SampleGames.Tarot Namespace Reference

Classes

- class SpreadManager
- class TarotSpread

5.7 CardHouse.TestScenes Namespace Reference

Classes

- class WaypointTesterCard
- class WaypointTesterGroup

5.8 CardHouse.Tutorial Namespace Reference

Classes

- class CardDragTutorial
- class ClosestCardHighlighter
- class DiscardAllCardsOperator
- class EventChainsTutorial
- class GridTutorial
- class GroupSetupTutorial
- class LaunchDataScriptable
- class MatureCropDragGate
- class MultiBoardTutorial
- class OutLinks
- class PhaseLabelUpdater
- class Plant
- class PlantGrowthScriptable
- class PlantMaturityInfo
- class PresentationPointTutorial
- class SandboxManager
- class SceneKeeper
- class SceneSpawner
- class SeekerTutorial
- class SplayTutorial
- class SpriteOperatorTutorial
- class SpriteVoterTutorial
- class StackTutorial
- class StringListScriptable
- class StringSeekerKVP
- class TransferOperatorTutorialUI
- class TutorialButton
- class ValidDragTutorial
- class WaterPlantAction
- class WaterTargetPlantGate

Chapter 6

Class Documentation

6.1 CardHouse.Activatable Class Reference

Inheritance diagram for CardHouse.Activatable:

Public Member Functions

- void Activate ()

Protected Member Functions

- virtual void OnActivate ()

6.1.1 Detailed Description

Definition at line 5 of file Activatable.cs.

6.1.2 Member Function Documentation

6.1.2.1 Activate()

```
void CardHouse.Activatable.Activate ( )
```

Definition at line 7 of file Activatable.cs.

6.1.2.2 OnActivate()

```
virtual void CardHouse.Activatable.OnActivate ( )  [protected], [virtual]
```

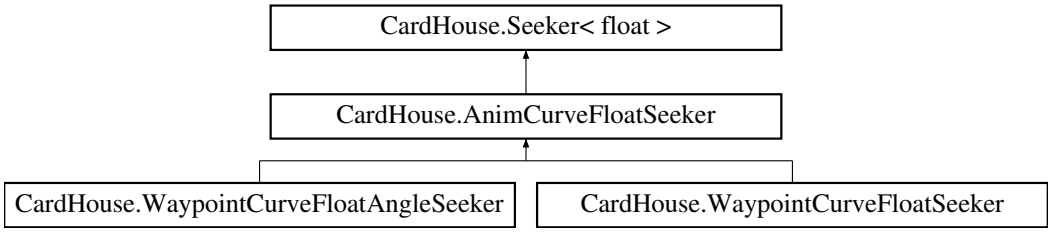
Definition at line 12 of file Activatable.cs.

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

- Activatable.cs

6.2 CardHouse.AnimCurveFloatSeeker Class Reference

Inheritance diagram for CardHouse.AnimCurveFloatSeeker:



Public Member Functions

- AnimCurveFloatSeeker (float duration, AnimationCurve progressCurve)
- override Seeker< float > MakeCopy ()
- override float Pump (float currentValue, float TimeSinceLastFrame)
- override bool IsDone (float currentValue)

Public Member Functions inherited from CardHouse.Seeker< float >

- abstract Seeker< T > **MakeCopy** ()
- void StartSeeking (T from, T to)
- abstract T **Pump** (T currentValue, float TimeSinceLastFrame)
- abstract bool **IsDone** (T currentValue)

Public Attributes

- float Duration
- AnimationCurve ProgressCurve

Public Attributes inherited from CardHouse.Seeker< float >

- T End

Protected Attributes

- float Timer

Protected Attributes inherited from CardHouse.Seeker< float >

- T Start

6.2.1 Detailed Description

Definition at line 5 of file AnimCurveFloatSeeker.cs.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 AnimCurveFloatSeeker()

```
CardHouse.AnimCurveFloatSeeker.AnimCurveFloatSeeker (
            float duration,
            AnimationCurve progressCurve )
```

Definition at line 11 of file AnimCurveFloatSeeker.cs.

6.2.3 Member Function Documentation

6.2.3.1 IsDone()

```
override bool CardHouse.AnimCurveFloatSeeker.IsDone (
            float currentValue )
```

Definition at line 30 of file AnimCurveFloatSeeker.cs.

6.2.3.2 MakeCopy()

```
override Seeker< float > CardHouse.AnimCurveFloatSeeker.MakeCopy ( )  [virtual]
```

Implements CardHouse.Seeker< float >.

Definition at line 18 of file AnimCurveFloatSeeker.cs.

6.2.3.3 Pump()

```
override float CardHouse.AnimCurveFloatSeeker.Pump (
            float currentValue,
            float TimeSinceLastFrame )
```

Definition at line 23 of file AnimCurveFloatSeeker.cs.

6.2.4 Member Data Documentation

6.2.4.1 Duration

```
float CardHouse.AnimCurveFloatSeeker.Duration
```

Definition at line 7 of file AnimCurveFloatSeeker.cs.

6.2.4.2 ProgressCurve

```
AnimationCurve CardHouse.AnimCurveFloatSeeker.ProgressCurve
```

Definition at line 9 of file AnimCurveFloatSeeker.cs.

6.2.4.3 Timer

```
float CardHouse.AnimCurveFloatSeeker.Timer  [protected]
```

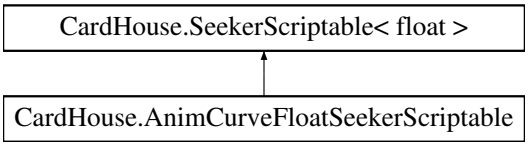
Definition at line 8 of file AnimCurveFloatSeeker.cs.

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

- AnimCurveFloatSeeker.cs

6.3 CardHouse.AnimCurveFloatSeekerScriptable Class Reference

Inheritance diagram for CardHouse.AnimCurveFloatSeekerScriptable:



Public Member Functions

- override Seeker< float > GetStrategy (params object[ ] args)
- abstract Seeker< T > **GetStrategy** (params object[ ] args)

Public Attributes

- float Duration = 2f
- AnimationCurve ProgressCurve

6.3.1 Detailed Description

Definition at line 6 of file AnimCurveFloatSeekerScriptable.cs.

6.3.2 Member Function Documentation

6.3.2.1 GetStrategy()

```
override Seeker< float > CardHouse.AnimCurveFloatSeekerScriptable.GetStrategy (
            params object[] args )  [virtual]
```

Implements CardHouse.SeekerScriptable< float >.

Definition at line 11 of file AnimCurveFloatSeekerScriptable.cs.

6.3.3 Member Data Documentation

6.3.3.1 Duration

```
float CardHouse.AnimCurveFloatSeekerScriptable.Duration = 2f
```

Definition at line 8 of file AnimCurveFloatSeekerScriptable.cs.

6.3.3.2 ProgressCurve

```
AnimationCurve CardHouse.AnimCurveFloatSeekerScriptable.ProgressCurve
```

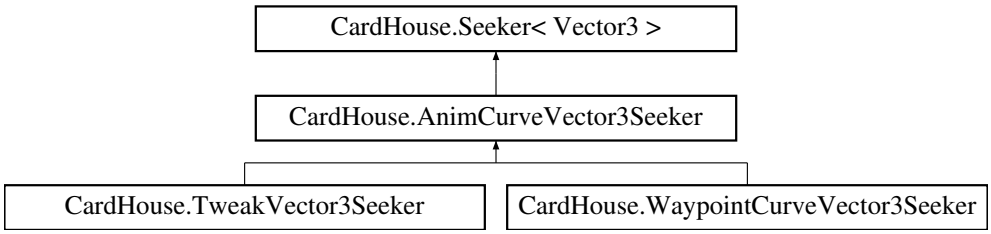
Definition at line 9 of file AnimCurveFloatSeekerScriptable.cs.

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

- AnimCurveFloatSeekerScriptable.cs

6.4 CardHouse.AnimCurveVector3Seeker Class Reference

Inheritance diagram for CardHouse.AnimCurveVector3Seeker:



Public Member Functions

- AnimCurveVector3Seeker (float duration, AnimationCurve progressCurve)
- override Seeker< Vector3 > MakeCopy ()
- override Vector3 Pump (Vector3 currentValue, float TimeSinceLastFrame)
- override bool IsDone (Vector3 currentValue)

Public Member Functions inherited from CardHouse.Seeker< Vector3 >

- abstract Seeker< T > **MakeCopy** ()
- void StartSeeking (T from, T to)
- abstract T **Pump** (T currentValue, float TimeSinceLastFrame)
- abstract bool **IsDone** (T currentValue)

Public Attributes

- float Duration
- AnimationCurve ProgressCurve

Public Attributes inherited from CardHouse.Seeker< Vector3 >

- T End

Protected Attributes

- float Timer

Protected Attributes inherited from CardHouse.Seeker< Vector3 >

- T Start

6.4.1 Detailed Description

Definition at line 5 of file AnimCurveVector3Seeker.cs.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 AnimCurveVector3Seeker()

```
CardHouse.AnimCurveVector3Seeker.AnimCurveVector3Seeker (
            float duration,
            AnimationCurve progressCurve )
```

Definition at line 11 of file AnimCurveVector3Seeker.cs.

6.4.3 Member Function Documentation

6.4.3.1 IsDone()

```
override bool CardHouse.AnimCurveVector3Seeker.IsDone (
            Vector3 currentValue )
```

Definition at line 30 of file AnimCurveVector3Seeker.cs.

6.4.3.2 MakeCopy()

```
override Seeker< Vector3 > CardHouse.AnimCurveVector3Seeker.MakeCopy ( )  [virtual]
```

Implements CardHouse.Seeker< Vector3 >.

Definition at line 18 of file AnimCurveVector3Seeker.cs.

6.4.3.3 Pump()

```
override Vector3 CardHouse.AnimCurveVector3Seeker.Pump (
            Vector3 currentValue,
            float TimeSinceLastFrame )
```

Definition at line 23 of file AnimCurveVector3Seeker.cs.

6.4.4 Member Data Documentation

6.4.4.1 Duration

```
float CardHouse.AnimCurveVector3Seeker.Duration
```

Definition at line 7 of file AnimCurveVector3Seeker.cs.

6.4.4.2 ProgressCurve

```
AnimationCurve CardHouse.AnimCurveVector3Seeker.ProgressCurve
```

Definition at line 9 of file AnimCurveVector3Seeker.cs.

6.4.4.3 Timer

```
float CardHouse.AnimCurveVector3Seeker.Timer  [protected]
```

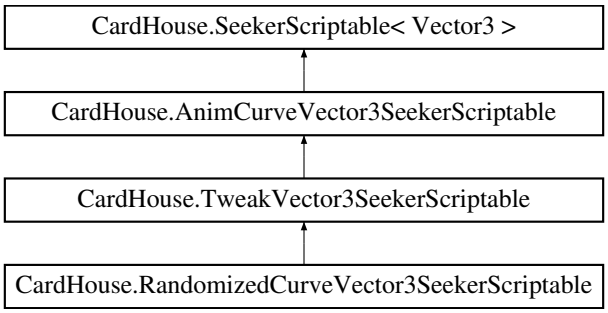
Definition at line 8 of file AnimCurveVector3Seeker.cs.

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

- AnimCurveVector3Seeker.cs

6.5 CardHouse.AnimCurveVector3SeekerScriptable Class Reference

Inheritance diagram for CardHouse.AnimCurveVector3SeekerScriptable:



Public Member Functions

- override Seeker< Vector3 > GetStrategy (params object[] args)
- abstract Seeker< T > **GetStrategy** (params object[] args)

Public Attributes

- float Duration = 2f
- AnimationCurve ProgressCurve

6.5.1 Detailed Description

Definition at line 6 of file AnimCurveVector3SeekerScriptable.cs.

6.5.2 Member Function Documentation

6.5.2.1 GetStrategy()

```
override Seeker< Vector3 > CardHouse.AnimCurveVector3SeekerScriptable.GetStrategy (
            params object[] args )  [virtual]
```

Implements CardHouse.SeekerScriptable< Vector3 >.

Definition at line 11 of file AnimCurveVector3SeekerScriptable.cs.

6.5.3 Member Data Documentation

6.5.3.1 Duration

```
float CardHouse.AnimCurveVector3SeekerScriptable.Duration = 2f
```

Definition at line 8 of file AnimCurveVector3SeekerScriptable.cs.

6.5.3.2 ProgressCurve

```
AnimationCurve CardHouse.AnimCurveVector3SeekerScriptable.ProgressCurve
```

Definition at line 9 of file AnimCurveVector3SeekerScriptable.cs.

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

- AnimCurveVector3SeekerScriptable.cs

6.6 CardHouse.ArcanaData Class Reference

Public Attributes

- MajorArcanaName Arcana
- TarotSuit Suit
- int Rank

6.6.1 Detailed Description

Definition at line 6 of file ArcanaData.cs.

6.6.2 Member Data Documentation

6.6.2.1 Arcana

```
MajorArcanaName CardHouse.ArcanaData.Arcana
```

Definition at line 8 of file ArcanaData.cs.

6.6.2.2 Rank

```
int CardHouse.ArcanaData.Rank
```

Definition at line 10 of file ArcanaData.cs.

6.6.2.3 Suit

```
TarotSuit CardHouse.ArcanaData.Suit
```

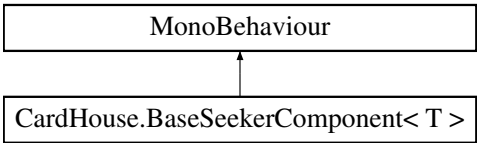
Definition at line 9 of file ArcanaData.cs.

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

- ArcanaData.cs

6.7 CardHouse.BaseSeekerComponent< T > Class Template Reference

Inheritance diagram for CardHouse.BaseSeekerComponent< T >:



Public Member Functions

- void StartSeeking (T destination, Seeker< T > strategy=null, bool useLocalSpace=false)

Public Attributes

- SeekerScriptable< T > Strategy

Protected Member Functions

- abstract Seeker< T > **GetDefaultSeeker** ()
- abstract T **GetCurrentValue** ()
- abstract void **SetNewValue** (T value)

Protected Attributes

- Seeker< T > MyStrategy
- bool IsSeeking
- bool UseLocalSpace

6.7.1 Detailed Description

Definition at line 5 of file BaseSeekerComponent.cs.

6.7.2 Member Function Documentation

6.7.2.1 StartSeeking()

```
void CardHouse.BaseSeekerComponent< T >.StartSeeking (
            T destination,
            Seeker< T > strategy = null,
            bool useLocalSpace = false )
```

Definition at line 19 of file BaseSeekerComponent.cs.

6.7.3 Member Data Documentation

6.7.3.1 IsSeeking

```
bool CardHouse.BaseSeekerComponent< T >.IsSeeking  [protected]
```

Definition at line 9 of file BaseSeekerComponent.cs.

6.7.3.2 MyStrategy

```
Seeker<T> CardHouse.BaseSeekerComponent< T >.MyStrategy  [protected]
```

Definition at line 7 of file BaseSeekerComponent.cs.

6.7.3.3 Strategy

```
SeekerScriptable<T> CardHouse.BaseSeekerComponent< T >.Strategy
```

Definition at line 12 of file BaseSeekerComponent.cs.

6.7.3.4 UseLocalSpace

```
bool CardHouse.BaseSeekerComponent< T >.UseLocalSpace  [protected]
```

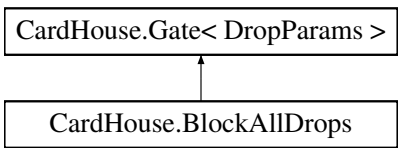
Definition at line 10 of file BaseSeekerComponent.cs.

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

- BaseSeekerComponent.cs

6.8 CardHouse.BlockAllDrops Class Reference

Inheritance diagram for CardHouse.BlockAllDrops:



Protected Member Functions

- override bool IsUnlockedInternal (DropParams gateParams)

Protected Member Functions inherited from CardHouse.Gate< DropParams >

- abstract bool **IsUnlockedInternal** (T argObject)

Additional Inherited Members

Public Member Functions inherited from CardHouse.Gate< DropParams >

- bool IsUnlocked (T argObject)

6.8.1 Detailed Description

Definition at line 6 of file BlockAllDrops.cs.

6.8.2 Member Function Documentation

6.8.2.1 IsUnlockedInternal()

```
override bool CardHouse.BlockAllDrops.IsUnlockedInternal (
            DropParams gateParams )  [protected]
```

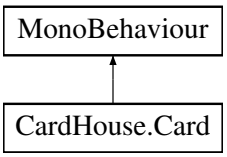
Definition at line 8 of file BlockAllDrops.cs.

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

- BlockAllDrops.cs

6.9 CardHouse.Card Class Reference

Inheritance diagram for CardHouse.Card:



Classes

- class GroupTransitionEvent

Public Member Functions

- void SetFacing (bool isFaceUp)
- void SetFacing (CardFacing facing, bool immediate=false, float spd=1f)
- void SetUpsideDown (bool isUpsideDown)
- void HandlePlayed ()
- CardGroup GetDiscardGroup ()
- void SetFocus (bool isFocused)
- void ToggleFocus ()
- void TriggerMountEvents (CardGroup group)
- void TriggerUnMountEvents (GroupName group)

Public Attributes

- CardGroup Group
- Animator FlipAnimator
- bool CanBeUpsideDown
- float UpsideDownChance = 0.5f
- Transform RootToRotateWhenUpsideDown
- Homing FaceHoming
- Turning FaceTurning
- Scaling FaceScaling
- List< GroupTransitionEvent > GroupTransitionEvents
- UnityEvent OnFlipUp
- UnityEvent OnFlipDown
- UnityEvent OnPlay
- Action< Card, CardGroup > OnMount

Static Public Attributes

- static Action< Card > OnCardFocused

Properties

- Homing Homing [get]
- Turning Turning [get]
- Scaling Scaling [get]
- CardFacing Facing [get]
- bool IsUpsideDown [get]

6.9.1 Detailed Description

Definition at line 9 of file Card.cs.

6.9.2 Member Function Documentation

6.9.2.1 GetDiscardGroup()

```
CardGroup CardHouse.Card.GetDiscardGroup ( )
```

Definition at line 113 of file Card.cs.

6.9.2.2 HandlePlayed()

```
void CardHouse.Card.HandlePlayed ( )
```

Definition at line 108 of file Card.cs.

6.9.2.3 SetFacing() [1/2]

```
void CardHouse.Card.SetFacing (
            bool isFaceUp )
```

Definition at line 71 of file Card.cs.

6.9.2.4 SetFacing() [2/2]

```
void CardHouse.Card.SetFacing (
            CardFacing facing,
            bool immediate = false,
            float spd = 1f )
```

Definition at line 76 of file Card.cs.

6.9.2.5 SetFocus()

```
void CardHouse.Card.SetFocus (
            bool isFocused )
```

Definition at line 124 of file Card.cs.

6.9.2.6 SetUpsideDown()

```
void CardHouse.Card.SetUpsideDown (
            bool isUpsideDown )
```

Definition at line 95 of file Card.cs.

6.9.2.7 ToggleFocus()

```
void CardHouse.Card.ToggleFocus ( )
```

Definition at line 144 of file Card.cs.

6.9.2.8 TriggerMountEvents()

```
void CardHouse.Card.TriggerMountEvents (
            CardGroup group )
```

Definition at line 149 of file Card.cs.

6.9.2.9 TriggerUnMountEvents()

```
void CardHouse.Card.TriggerUnMountEvents (
            GroupName group )
```

Definition at line 164 of file Card.cs.

6.9.3 Member Data Documentation

6.9.3.1 CanBeUpsideDown

```
bool CardHouse.Card.CanBeUpsideDown
```

Definition at line 27 of file Card.cs.

6.9.3.2 FaceHoming

```
Homing CardHouse.Card.FaceHoming
```

Definition at line 32 of file Card.cs.

6.9.3.3 FaceScaling

```
Scaling CardHouse.Card.FaceScaling
```

Definition at line 34 of file Card.cs.

6.9.3.4 FaceTurning

```
Turning CardHouse.Card.FaceTurning
```

Definition at line 33 of file Card.cs.

6.9.3.5 FlipAnimator

```
Animator CardHouse.Card.FlipAnimator
```

Definition at line 25 of file Card.cs.

6.9.3.6 Group

```
CardGroup CardHouse.Card.Group
```

Definition at line 20 of file Card.cs.

6.9.3.7 GroupTransitionEvents

```
List<GroupTransitionEvent> CardHouse.Card.GroupTransitionEvents
```

Definition at line 36 of file Card.cs.

6.9.3.8 OnCardFocused

```
Action<Card> CardHouse.Card.OnCardFocused  [static]
```

Definition at line 48 of file Card.cs.

6.9.3.9 OnFlipDown

```
UnityEvent CardHouse.Card.OnFlipDown
```

Definition at line 41 of file Card.cs.

6.9.3.10 OnFlipUp

```
UnityEvent CardHouse.Card.OnFlipUp
```

Definition at line 40 of file Card.cs.

6.9.3.11 OnMount

```
Action<Card, CardGroup> CardHouse.Card.OnMount
```

Definition at line 44 of file Card.cs.

6.9.3.12 OnPlay

```
UnityEvent CardHouse.Card.OnPlay
```

Definition at line 42 of file Card.cs.

6.9.3.13 RootToRotateWhenUpsideDown

```
Transform CardHouse.Card.RootToRotateWhenUpsideDown
```

Definition at line 30 of file Card.cs.

6.9.3.14 UpsideDownChance

```
float CardHouse.Card.UpsideDownChance = 0.5f
```

Definition at line 29 of file Card.cs.

6.9.4 Property Documentation

6.9.4.1 Facing

```
CardFacing CardHouse.Card.Facing  [get]
```

Definition at line 38 of file Card.cs.

6.9.4.2 Homing

```
Homing CardHouse.Card.Homing  [get]
```

Definition at line 21 of file Card.cs.

6.9.4.3 IsUpsideDown

```
bool CardHouse.Card.IsUpsideDown  [get]
```

Definition at line 106 of file Card.cs.

6.9.4.4 Scaling

```
Scaling CardHouse.Card.Scaling  [get]
```

Definition at line 23 of file Card.cs.

6.9.4.5 Turning

```
Turning CardHouse.Card.Turning  [get]
```

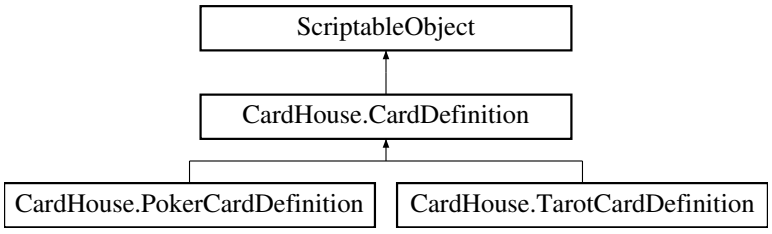
Definition at line 22 of file Card.cs.

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

- Card.cs

6.10 CardHouse.CardDefinition Class Reference

Inheritance diagram for CardHouse.CardDefinition:



Public Attributes

- Sprite BackArt

6.10.1 Detailed Description

Definition at line 5 of file CardDefinition.cs.

6.10.2 Member Data Documentation

6.10.2.1 BackArt

```
Sprite CardHouse.CardDefinition.BackArt
```

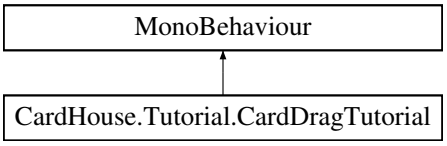
Definition at line 7 of file CardDefinition.cs.

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

- CardDefinition.cs

6.11 CardHouse.Tutorial.CardDragTutorial Class Reference

Inheritance diagram for CardHouse.Tutorial.CardDragTutorial:



Public Member Functions

- void AdjustDragSwellSlider ()
- void AdjustSeekerGainSlider ()
- void OnGrabOffsetToggled ()
- void AdjustOffsetX ()
- void AdjustOffsetY ()
- void ShowSwellOutline ()

Public Attributes

- Slider DragSwellSlider
- TMP_Text DragSwellText
- Slider SeekerGainSlider
- TMP_Text SeekerGainText
- Toggle GrabOffsetToggle
- Slider XOffsetSlider
- TMP_Text XOffsetText
- Slider YOffsetSlider
- TMP_Text YOffsetText
- Card Card

6.11.1 Detailed Description

Definition at line 8 of file CardDragTutorial.cs.

6.11.2 Member Function Documentation

6.11.2.1 AdjustDragSwellSlider()

```
void CardHouse.Tutorial.CardDragTutorial.AdjustDragSwellSlider ( )
```

Definition at line 30 of file CardDragTutorial.cs.

6.11.2.2 AdjustOffsetX()

```
void CardHouse.Tutorial.CardDragTutorial.AdjustOffsetX ( )
```

Definition at line 61 of file CardDragTutorial.cs.

6.11.2.3 AdjustOffsetY()

```
void CardHouse.Tutorial.CardDragTutorial.AdjustOffsetY ( )
```

Definition at line 68 of file CardDragTutorial.cs.

6.11.2.4 AdjustSeekerGainSlider()

```
void CardHouse.Tutorial.CardDragTutorial.AdjustSeekerGainSlider ( )
```

Definition at line 40 of file CardDragTutorial.cs.

6.11.2.5 OnGrabOffsetToggled()

```
void CardHouse.Tutorial.CardDragTutorial.OnGrabOffsetToggled ( )
```

Definition at line 47 of file CardDragTutorial.cs.

6.11.2.6 ShowSwellOutline()

```
void CardHouse.Tutorial.CardDragTutorial.ShowSwellOutline ( )
```

Definition at line 81 of file CardDragTutorial.cs.

6.11.3 Member Data Documentation

6.11.3.1 Card

```
Card CardHouse.Tutorial.CardDragTutorial.Card
```

Definition at line 20 of file CardDragTutorial.cs.

6.11.3.2 DragSwellSlider

```
Slider CardHouse.Tutorial.CardDragTutorial.DragSwellSlider
```

Definition at line 10 of file CardDragTutorial.cs.

6.11.3.3 DragSwellText

```
TMP_Text CardHouse.Tutorial.CardDragTutorial.DragSwellText
```

Definition at line 11 of file CardDragTutorial.cs.

6.11.3.4 GrabOffsetToggle

```
Toggle CardHouse.Tutorial.CardDragTutorial.GrabOffsetToggle
```

Definition at line 14 of file CardDragTutorial.cs.

6.11.3.5 SeekerGainSlider

```
Slider CardHouse.Tutorial.CardDragTutorial.SeekerGainSlider
```

Definition at line 12 of file CardDragTutorial.cs.

6.11.3.6 SeekerGainText

```
TMP_Text CardHouse.Tutorial.CardDragTutorial.SeekerGainText
```

Definition at line 13 of file CardDragTutorial.cs.

6.11.3.7 XOffsetSlider

```
Slider CardHouse.Tutorial.CardDragTutorial.XOffsetSlider
```

Definition at line 15 of file CardDragTutorial.cs.

6.11.3.8 XOffsetText

```
TMP_Text CardHouse.Tutorial.CardDragTutorial.XOffsetText
```

Definition at line 16 of file CardDragTutorial.cs.

6.11.3.9 YOffsetSlider

```
Slider CardHouse.Tutorial.CardDragTutorial.YOffsetSlider
```

Definition at line 17 of file CardDragTutorial.cs.

6.11.3.10 YOffsetText

```
TMP_Text CardHouse.Tutorial.CardDragTutorial.YOffsetText
```

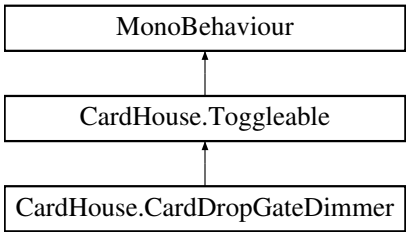
Definition at line 18 of file CardDragTutorial.cs.

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

- CardDragTutorial.cs

6.12 CardHouse.CardDropGateDimmer Class Reference

Inheritance diagram for CardHouse.CardDropGateDimmer:



Public Attributes

- MultiSpriteOperator Handler
- string ActiveMessage
- string InactiveMessage

Public Attributes inherited from CardHouse.Toggleable

- bool IsActive = true

Additional Inherited Members

Public Member Functions inherited from CardHouse.Toggleable

- void SetIsActive (bool newValue)

6.12.1 Detailed Description

Definition at line 6 of file CardDropGateDimmer.cs.

6.12.2 Member Data Documentation

6.12.2.1 ActiveMessage

```
string CardHouse.CardDropGateDimmer.ActiveMessage
```

Definition at line 9 of file CardDropGateDimmer.cs.

6.12.2.2 Handler

```
MultiSpriteOperator CardHouse.CardDropGateDimmer.Handler
```

Definition at line 8 of file CardDropGateDimmer.cs.

6.12.2.3 InactiveMessage

```
string CardHouse.CardDropGateDimmer.InactiveMessage
```

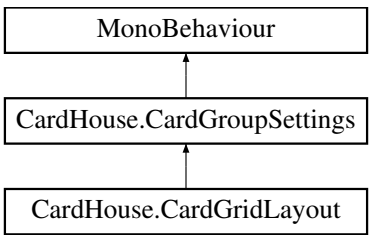
Definition at line 10 of file CardDropGateDimmer.cs.

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

- CardDropGateDimmer.cs

6.13 CardHouse.CardGridLayout Class Reference

Inheritance diagram for CardHouse.CardGridLayout:



Public Attributes

- int CardsPerRow = 5
- float MarginalCardOffset = 0.05f
- bool Straighten = true

Public Attributes inherited from CardHouse.CardGroupSettings

- int CardLimit = -1
- float MountedCardAltitude = 0.01f
- CardFacing ForcedFacing
- GroupInteractability ForcedInteractability
- MountingMode DragMountingMode = MountingMode.Top
- bool UseMyScale = false

Protected Member Functions

- override void ApplySpacing (List< Card > cards, SeekerSetList seekerSets)

- abstract void **ApplySpacing** (List< Card > cards, SeekerSetList seekerSets)

Additional Inherited Members

Public Member Functions inherited from CardHouse.CardGroupSettings

- void Apply (List< Card > cards, bool instaFlip=false, SeekerSetList seekerSets=null)

6.13.1 Detailed Description

Definition at line 6 of file CardGridLayout.cs.

6.13.2 Member Function Documentation

6.13.2.1 ApplySpacing()

```
override void CardHouse.CardGridLayout.ApplySpacing (
            List< Card > cards,
            SeekerSetList seekerSets )  [protected], [virtual]
```

Implements CardHouse.CardGroupSettings.

Definition at line 22 of file CardGridLayout.cs.

6.13.3 Member Data Documentation

6.13.3.1 CardsPerRow

```
int CardHouse.CardGridLayout.CardsPerRow = 5
```

Definition at line 8 of file CardGridLayout.cs.

6.13.3.2 MarginalCardOffset

```
float CardHouse.CardGridLayout.MarginalCardOffset = 0.05f
```

Definition at line 9 of file CardGridLayout.cs.

6.13.3.3 Straighten

```
bool CardHouse.CardGridLayout.Straighten = true
```

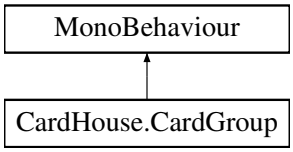
Definition at line 11 of file CardGridLayout.cs.

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

- CardGridLayout.cs

6.14 CardHouse.CardGroup Class Reference

Inheritance diagram for CardHouse.CardGroup:



Public Member Functions

- int? GetClosestMountedCardIndex (Vector3 position)
- void HandleTriggerEnter2D (Collider2D col)
- void HandleTriggerExit2D (Collider2D col)
- void SetHilightState (bool newState)
- void ApplyStrategy ()
- bool HasRoom ()
- void Mount (Card card, int? index=null, bool instaFlip=false, SeekerSetList seekerSets=null, SeekerSet seekersForUnmounting=null)
- bool SafeMount (Card card, int? index=null)
- int? UnMount (Card card, SeekerSet seekersForUnmounting=null)
- Card UnMount (int? index=null, SeekerSet seekersForUnmounting=null)
- Card Get (int? index=null)
- List< Card > Get (GroupTargetType targetType, int count)
- int? IndexOf (Card card)
- void Shuffle (bool isInstant=false)
- void ShuffleIn (List< Card > cards, bool isInstant=false)

Static Public Member Functions

- static void AddHoveredGroup (CardGroup group)
- static void RemoveHoveredGroup (CardGroup group)
- static Card GetActiveCard (DragDetector draggable)

Public Attributes

- bool HilightOnCardEntry = true
- GameObject Hilight
- GateCollection< DropParams > DropGates
- SeekerScriptable< Vector3 > ShuffleStrategy
- List< Card > MountedCards = new List<Card>()
- UnityEvent OnGroupChanged

Static Public Attributes

- static Action< CardGroup > OnNewActiveGroup
- static Action< Card, Card > OnCardUsedOnTarget

Properties

- static CardGroup HilightedGroup `[get]`

6.14.1 Detailed Description

Definition at line 10 of file CardGroup.cs.

6.14.2 Member Function Documentation

6.14.2.1 AddHoveredGroup()

```
static void CardHouse.CardGroup.AddHoveredGroup (
            CardGroup group ) [static]
```

Definition at line 39 of file CardGroup.cs.

6.14.2.2 ApplyStrategy()

```
void CardHouse.CardGroup.ApplyStrategy ( )
```

Definition at line 333 of file CardGroup.cs.

6.14.2.3 Get() [1/2]

```
List< Card > CardHouse.CardGroup.Get (
            GroupTargetType targetType,
            int count )
```

Definition at line 426 of file CardGroup.cs.

6.14.2.4 Get() [2/2]

```
Card CardHouse.CardGroup.Get (
            int?  index = null )
```

Definition at line 404 of file CardGroup.cs.

6.14.2.5 GetActiveCard()

```
static Card CardHouse.CardGroup.GetActiveCard (
            DragDetector draggable )  [static]
```

Definition at line 57 of file CardGroup.cs.

6.14.2.6 GetClosestMountedCardIndex()

```
int?  CardHouse.CardGroup.GetClosestMountedCardIndex (
            Vector3 position )
```

Definition at line 241 of file CardGroup.cs.

6.14.2.7 HandleTriggerEnter2D()

```
void CardHouse.CardGroup.HandleTriggerEnter2D (
            Collider2D col )
```

Definition at line 270 of file CardGroup.cs.

6.14.2.8 HandleTriggerExit2D()

```
void CardHouse.CardGroup.HandleTriggerExit2D (
            Collider2D col )
```

Definition at line 284 of file CardGroup.cs.

6.14.2.9 HasRoom()

```
bool CardHouse.CardGroup.HasRoom ( )
```

Definition at line 338 of file CardGroup.cs.

6.14.2.10 IndexOf()

```
int?  CardHouse.CardGroup.IndexOf (
            Card card )
```

Definition at line 460 of file CardGroup.cs.

6.14.2.11 Mount()

```
void CardHouse.CardGroup.Mount (
            Card card,
            int?  index = null,
            bool instaFlip = false,
            SeekerSetList seekerSets = null,
            SeekerSet seekersForUnmounting = null )
```

Definition at line 343 of file CardGroup.cs.

6.14.2.12 RemoveHoveredGroup()

```
static void CardHouse.CardGroup.RemoveHoveredGroup (
            CardGroup group )  [static]
```

Definition at line 46 of file CardGroup.cs.

6.14.2.13 SafeMount()

```
bool CardHouse.CardGroup.SafeMount (
            Card card,
            int?  index = null )
```

Definition at line 367 of file CardGroup.cs.

6.14.2.14 SetHilightState()

```
void CardHouse.CardGroup.SetHilightState (
            bool newState )
```

Definition at line 325 of file CardGroup.cs.

6.14.2.15 Shuffle()

```
void CardHouse.CardGroup.Shuffle (
            bool isInstant = false )
```

Definition at line 469 of file CardGroup.cs.

6.14.2.16 ShuffleIn()

```
void CardHouse.CardGroup.ShuffleIn (
            List< Card > cards,
            bool isInstant = false )
```

Definition at line 504 of file CardGroup.cs.

6.14.2.17 UnMount() [1/2]

```
int?  CardHouse.CardGroup.UnMount (
            Card card,
            SeekerSet seekersForUnmounting = null )
```

Definition at line 377 of file CardGroup.cs.

6.14.2.18 UnMount() [2/2]

```
Card CardHouse.CardGroup.UnMount (
            int?  index = null,
            SeekerSet seekersForUnmounting = null )
```

Definition at line 389 of file CardGroup.cs.

6.14.3 Member Data Documentation

6.14.3.1 DropGates

```
GateCollection<DropParams> CardHouse.CardGroup.DropGates
```

Definition at line 15 of file CardGroup.cs.

6.14.3.2 Hilight

```
GameObject CardHouse.CardGroup.Hilight
```

Definition at line 13 of file CardGroup.cs.

6.14.3.3 HilightOnCardEntry

```
bool CardHouse.CardGroup.HilightOnCardEntry = true
```

Definition at line 12 of file CardGroup.cs.

6.14.3.4 MountedCards

```
List<Card> CardHouse.CardGroup.MountedCards = new List<Card>()
```

Definition at line 19 of file CardGroup.cs.

6.14.3.5 OnCardUsedOnTarget

```
Action<Card, Card> CardHouse.CardGroup.OnCardUsedOnTarget  [static]
```

Definition at line 25 of file CardGroup.cs.

6.14.3.6 OnGroupChanged

```
UnityEvent CardHouse.CardGroup.OnGroupChanged
```

Definition at line 22 of file CardGroup.cs.

6.14.3.7 OnNewActiveGroup

```
Action<CardGroup> CardHouse.CardGroup.OnNewActiveGroup  [static]
```

Definition at line 24 of file CardGroup.cs.

6.14.3.8 ShuffleStrategy

```
SeekerScriptable<Vector3> CardHouse.CardGroup.ShuffleStrategy
```

Definition at line 17 of file CardGroup.cs.

6.14.4 Property Documentation

6.14.4.1 HilightedGroup

```
CardGroup CardHouse.CardGroup.HilightedGroup  [static], [get]
```

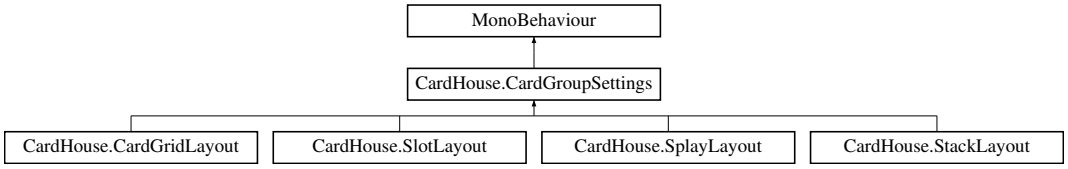
Definition at line 31 of file CardGroup.cs.

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

- CardGroup.cs

6.15 CardHouse.CardGroupSettings Class Reference

Inheritance diagram for CardHouse.CardGroupSettings:



Public Member Functions

- void Apply (List< Card > cards, bool instaFlip=false, SeekerSetList seekerSets=null)

Public Attributes

- int CardLimit = -1
- float MountedCardAltitude = 0.01f
- CardFacing ForcedFacing
- GroupInteractability ForcedInteractability
- MountingMode DragMountingMode = MountingMode.Top
- bool UseMyScale = false

Protected Member Functions

- abstract void **ApplySpacing** (List< Card > cards, SeekerSetList seekerSets)

6.15.1 Detailed Description

Definition at line 6 of file CardGroupSettings.cs.

6.15.2 Member Function Documentation

6.15.2.1 Apply()

```
void CardHouse.CardGroupSettings.Apply (
            List< Card > cards,
            bool instaFlip = false,
            SeekerSetList seekerSets = null )
```

Definition at line 15 of file CardGroupSettings.cs.

6.15.3 Member Data Documentation

6.15.3.1 CardLimit

```
int CardHouse.CardGroupSettings.CardLimit = -1
```

Definition at line 8 of file CardGroupSettings.cs.

6.15.3.2 DragMountingMode

```
MountingMode CardHouse.CardGroupSettings.DragMountingMode = MountingMode.Top
```

Definition at line 12 of file CardGroupSettings.cs.

6.15.3.3 ForcedFacing

```
CardFacing CardHouse.CardGroupSettings.ForcedFacing
```

Definition at line 10 of file CardGroupSettings.cs.

6.15.3.4 ForcedInteractability

```
GroupInteractability CardHouse.CardGroupSettings.ForcedInteractability
```

Definition at line 11 of file CardGroupSettings.cs.

6.15.3.5 MountedCardAltitude

```
float CardHouse.CardGroupSettings.MountedCardAltitude = 0.01f
```

Definition at line 9 of file CardGroupSettings.cs.

6.15.3.6 UseMyScale

```
bool CardHouse.CardGroupSettings.UseMyScale = false
```

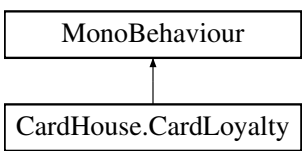
Definition at line 13 of file CardGroupSettings.cs.

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

- CardGroupSettings.cs

6.16 CardHouse.CardLoyalty Class Reference

Inheritance diagram for CardHouse.CardLoyalty:



Public Attributes

- int PlayerIndex

6.16.1 Detailed Description

Definition at line 5 of file CardLoyalty.cs.

6.16.2 Member Data Documentation

6.16.2.1 PlayerIndex

```
int CardHouse.CardLoyalty.PlayerIndex
```

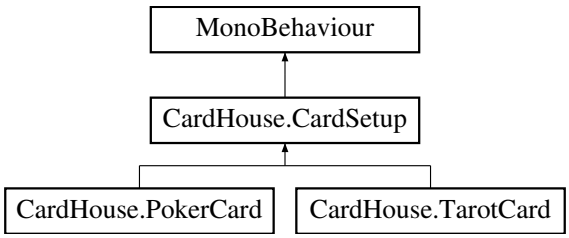
Definition at line 7 of file CardLoyalty.cs.

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

- CardLoyalty.cs

6.17 CardHouse.CardSetup Class Reference

Inheritance diagram for CardHouse.CardSetup:



Public Member Functions

- abstract void **Apply** (CardDefinition data)

6.17.1 Detailed Description

Definition at line 5 of file CardSetup.cs.

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

- CardSetup.cs

6.18 CardHouse.CardTargetCardOperator Class Reference

Inheritance diagram for CardHouse.CardTargetCardOperator:

Public Attributes

- SeekerScriptableSet DiscardSeekers

Protected Member Functions

- override void OnActivate ()
- abstract void **ActOnTarget** ()

- virtual void OnActivate ()

Protected Attributes

- Card MyCard
- Card Target

Additional Inherited Members

Public Member Functions inherited from CardHouse.Activatable

- void Activate ()

6.18.1 Detailed Description

Definition at line 6 of file CardTargetCardOperator.cs.

6.18.2 Member Function Documentation

6.18.2.1 OnActivate()

```
override void CardHouse.CardTargetCardOperator.OnActivate ( )  [protected], [virtual]
```

Reimplemented from CardHouse.Activatable.

Definition at line 31 of file CardTargetCardOperator.cs.

6.18.3 Member Data Documentation

6.18.3.1 DiscardSeekers

```
SeekerScriptableSet CardHouse.CardTargetCardOperator.DiscardSeekers
```

Definition at line 8 of file CardTargetCardOperator.cs.

6.18.3.2 MyCard

```
Card CardHouse.CardTargetCardOperator.MyCard  [protected]
```

Definition at line 9 of file CardTargetCardOperator.cs.

6.18.3.3 Target

```
Card CardHouse.CardTargetCardOperator.Target  [protected]
```

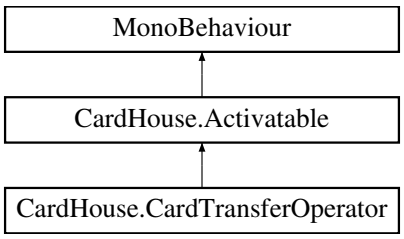
Definition at line 10 of file CardTargetCardOperator.cs.

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

- CardTargetCardOperator.cs

6.19 CardHouse.CardTransferOperator Class Reference

Inheritance diagram for CardHouse.CardTransferOperator:



Public Attributes

- GroupTransition Transition
- GroupTargetType GrabFrom = GroupTargetType.Last
- GroupTargetType SendTo = GroupTargetType.Last
- int NumberToTransfer = 1
- float FlipSpeed = 1
- SeekerScriptable< Vector3 > PopPushHomingOverride
- List< TimedEvent > OnSourceDepletedEventChain
- bool TryAgainAfterSourceDepleted

Protected Member Functions

- override void OnActivate ()
- virtual void OnActivate ()

Additional Inherited Members

Public Member Functions inherited from CardHouse.Activatable

- void Activate ()

6.19.1 Detailed Description

Definition at line 8 of file CardTransferOperator.cs.

6.19.2 Member Function Documentation

6.19.2.1 OnActivate()

```
override void CardHouse.CardTransferOperator.OnActivate ( )  [protected], [virtual]
```

Reimplemented from CardHouse.Activatable.

Definition at line 22 of file CardTransferOperator.cs.

6.19.3 Member Data Documentation

6.19.3.1 FlipSpeed

```
float CardHouse.CardTransferOperator.FlipSpeed = 1
```

Definition at line 14 of file CardTransferOperator.cs.

6.19.3.2 GrabFrom

```
GroupTargetType CardHouse.CardTransferOperator.GrabFrom = GroupTargetType.Last
```

Definition at line 11 of file CardTransferOperator.cs.

6.19.3.3 NumberToTransfer

```
int CardHouse.CardTransferOperator.NumberToTransfer = 1
```

Definition at line 13 of file CardTransferOperator.cs.

6.19.3.4 OnSourceDepletedEventChain

```
List<TimedEvent> CardHouse.CardTransferOperator.OnSourceDepletedEventChain
```

Definition at line 18 of file CardTransferOperator.cs.

6.19.3.5 PopPushHomingOverride

```
SeekerScriptable<Vector3> CardHouse.CardTransferOperator.PopPushHomingOverride
```

Definition at line 16 of file CardTransferOperator.cs.

6.19.3.6 SendTo

```
GroupTargetType CardHouse.CardTransferOperator.SendTo = GroupTargetType.Last
```

Definition at line 12 of file CardTransferOperator.cs.

6.19.3.7 Transition

```
GroupTransition CardHouse.CardTransferOperator.Transition
```

Definition at line 10 of file CardTransferOperator.cs.

6.19.3.8 TryAgainAfterSourceDepleted

```
bool CardHouse.CardTransferOperator.TryAgainAfterSourceDepleted
```

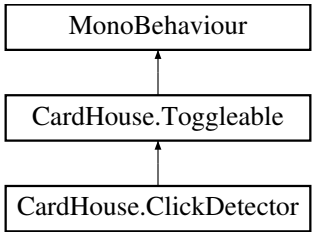
Definition at line 20 of file CardTransferOperator.cs.

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

- CardTransferOperator.cs

6.20 CardHouse.ClickDetector Class Reference

Inheritance diagram for CardHouse.ClickDetector:



Public Attributes

- UnityEvent OnPress
- UnityEvent OnButtonClicked
- GateCollection< NoParams > ClickGates

Public Attributes inherited from CardHouse.Toggleable

- bool IsActive = true

Additional Inherited Members

Public Member Functions inherited from CardHouse.Toggleable

- void SetIsActive (bool newValue)

6.20.1 Detailed Description

Definition at line 5 of file ClickDetector.cs.

6.20.2 Member Data Documentation

6.20.2.1 ClickGates

```
GateCollection<NoParams> CardHouse.ClickDetector.ClickGates
```

Definition at line 10 of file ClickDetector.cs.

6.20.2.2 OnButtonClicked

```
UnityEvent CardHouse.ClickDetector.OnButtonClicked
```

Definition at line 8 of file ClickDetector.cs.

6.20.2.3 OnPress

```
UnityEvent CardHouse.ClickDetector.OnPress
```

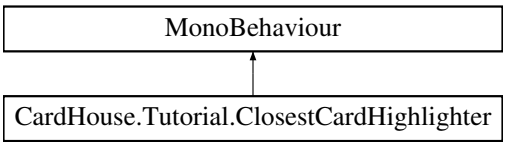
Definition at line 7 of file ClickDetector.cs.

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

- ClickDetector.cs

6.21 CardHouse.Tutorial.ClosestCardHighlighter Class Reference

Inheritance diagram for CardHouse.Tutorial.ClosestCardHighlighter:



6.21.1 Detailed Description

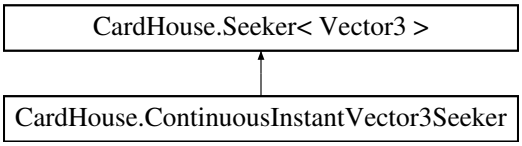
Definition at line 5 of file ClosestCardHighlighter.cs.

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

- ClosestCardHighlighter.cs

6.22 CardHouse.ContinuousInstantVector3Seeker Class Reference

Inheritance diagram for CardHouse.ContinuousInstantVector3Seeker:



Public Member Functions

- override Seeker< Vector3 > MakeCopy ()
- override Vector3 Pump (Vector3 currentValue, float TimeSinceLastFrame)
- override bool IsDone (Vector3 currentValue)

Public Member Functions inherited from CardHouse.Seeker< Vector3 >

- abstract Seeker< T > **MakeCopy** ()
- void StartSeeking (T from, T to)
- abstract T **Pump** (T currentValue, float TimeSinceLastFrame)
- abstract bool **IsDone** (T currentValue)

Additional Inherited Members

Public Attributes inherited from CardHouse.Seeker< Vector3 >

- T End

Protected Attributes inherited from CardHouse.Seeker< Vector3 >

- T Start

6.22.1 Detailed Description

Definition at line 5 of file ContinuousInstantVector3Seeker.cs.

6.22.2 Member Function Documentation

6.22.2.1 IsDone()

```
override bool CardHouse.ContinuousInstantVector3Seeker.IsDone (
            Vector3 currentValue )
```

Definition at line 17 of file ContinuousInstantVector3Seeker.cs.

6.22.2.2 MakeCopy()

```
override Seeker< Vector3 > CardHouse.ContinuousInstantVector3Seeker.MakeCopy ( )  [virtual]
```

Implements CardHouse.Seeker< Vector3 >.

Definition at line 7 of file ContinuousInstantVector3Seeker.cs.

6.22.2.3 Pump()

```
override Vector3 CardHouse.ContinuousInstantVector3Seeker.Pump (
            Vector3 currentValue,
            float TimeSinceLastFrame )
```

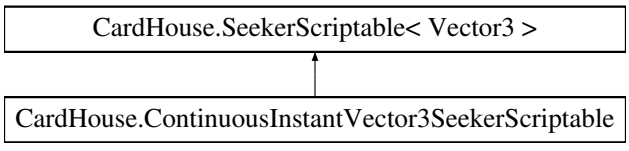
Definition at line 12 of file ContinuousInstantVector3Seeker.cs.

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

- ContinuousInstantVector3Seeker.cs

6.23 CardHouse.ContinuousInstantVector3SeekerScriptable Class Reference

Inheritance diagram for CardHouse.ContinuousInstantVector3SeekerScriptable:



Public Member Functions

- override Seeker< Vector3 > GetStrategy (params object[] args)
- abstract Seeker< T > **GetStrategy** (params object[] args)

6.23.1 Detailed Description

Definition at line 6 of file ContinuousInstantVector3SeekerScriptable.cs.

6.23.2 Member Function Documentation

6.23.2.1 GetStrategy()

```
override Seeker< Vector3 > CardHouse.ContinuousInstantVector3SeekerScriptable.GetStrategy (
            params object[] args )  [virtual]
```

Implements CardHouse.SeekerScriptable< Vector3 >.

Definition at line 8 of file ContinuousInstantVector3SeekerScriptable.cs.

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

- ContinuousInstantVector3SeekerScriptable.cs

6.24 CardHouse.CurrencyCost.CostWithLabel Class Reference

Public Attributes

- CurrencyQuantity Cost
- TextMeshPro Label

6.24.1 Detailed Description

Definition at line 11 of file CurrencyCost.cs.

6.24.2 Member Data Documentation

6.24.2.1 Cost

```
CurrencyQuantity CardHouse.CurrencyCost.CostWithLabel.Cost
```

Definition at line 13 of file CurrencyCost.cs.

6.24.2.2 Label

```
TextMeshPro CardHouse.CurrencyCost.CostWithLabel.Label
```

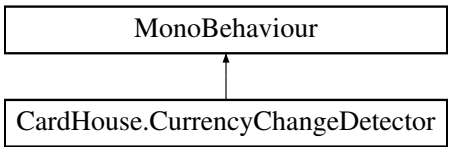
Definition at line 14 of file CurrencyCost.cs.

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

- CurrencyCost.cs

6.25 CardHouse.CurrencyChangeDetector Class Reference

Inheritance diagram for CardHouse.CurrencyChangeDetector:



Public Attributes

- UnityEvent OnCurrencyChange

6.25.1 Detailed Description

Definition at line 7 of file CurrencyChangeDetector.cs.

6.25.2 Member Data Documentation

6.25.2.1 OnCurrencyChange

```
UnityEvent CardHouse.CurrencyChangeDetector.OnCurrencyChange
```

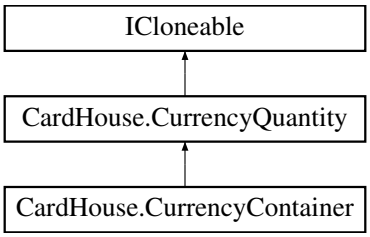
Definition at line 10 of file CurrencyChangeDetector.cs.

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

- CurrencyChangeDetector.cs

6.26 CardHouse.CurrencyContainer Class Reference

Inheritance diagram for CardHouse.CurrencyContainer:



Public Member Functions

- void Adjust (int amount)
- override object Clone ()

- virtual object Clone ()

Public Attributes

- bool HasMax
- int Max
- bool HasMin = true
- int Min
- int RefillValue

Public Attributes inherited from CardHouse.CurrencyQuantity

- CurrencyScriptable CurrencyType
- int Amount

6.26.1 Detailed Description

Definition at line 6 of file CurrencyContainer.cs.

6.26.2 Member Function Documentation

6.26.2.1 Adjust()

```
void CardHouse.CurrencyContainer.Adjust (
            int amount )
```

Definition at line 14 of file CurrencyContainer.cs.

6.26.2.2 Clone()

```
override object CardHouse.CurrencyContainer.Clone ( )  [virtual]
```

Reimplemented from CardHouse.CurrencyQuantity.

Definition at line 27 of file CurrencyContainer.cs.

6.26.3 Member Data Documentation

6.26.3.1 HasMax

```
bool CardHouse.CurrencyContainer.HasMax
```

Definition at line 8 of file CurrencyContainer.cs.

6.26.3.2 HasMin

```
bool CardHouse.CurrencyContainer.HasMin = true
```

Definition at line 10 of file CurrencyContainer.cs.

6.26.3.3 Max

```
int CardHouse.CurrencyContainer.Max
```

Definition at line 9 of file CurrencyContainer.cs.

6.26.3.4 Min

```
int CardHouse.CurrencyContainer.Min
```

Definition at line 11 of file CurrencyContainer.cs.

6.26.3.5 RefillValue

```
int CardHouse.CurrencyContainer.RefillValue
```

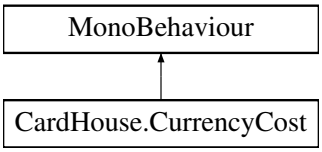
Definition at line 12 of file CurrencyContainer.cs.

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

- CurrencyContainer.cs

6.27 CardHouse.CurrencyCost Class Reference

Inheritance diagram for CardHouse.CurrencyCost:



Classes

- class CostWithLabel

Public Member Functions

- void Activate ()

Public Attributes

- List< CostWithLabel > Cost

6.27.1 Detailed Description

Definition at line 8 of file CurrencyCost.cs.

6.27.2 Member Function Documentation

6.27.2.1 Activate()

```
void CardHouse.CurrencyCost.Activate ( )
```

Definition at line 30 of file CurrencyCost.cs.

6.27.3 Member Data Documentation

6.27.3.1 Cost

```
List<CostWithLabel> CardHouse.CurrencyCost.Cost
```

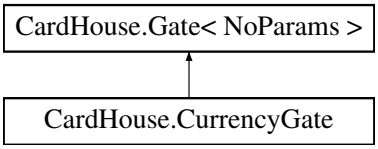
Definition at line 17 of file CurrencyCost.cs.

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

- CurrencyCost.cs

6.28 CardHouse.CurrencyGate Class Reference

Inheritance diagram for CardHouse.CurrencyGate:



Protected Member Functions

- override bool IsUnlockedInternal (NoParams gateParams)

Protected Member Functions inherited from CardHouse.Gate< NoParams >

- abstract bool **IsUnlockedInternal** (T argObject)

Additional Inherited Members

Public Member Functions inherited from CardHouse.Gate< NoParams >

- bool IsUnlocked (T argObject)

6.28.1 Detailed Description

Definition at line 6 of file CurrencyGate.cs.

6.28.2 Member Function Documentation

6.28.2.1 IsUnlockedInternal()

```
override bool CardHouse.CurrencyGate.IsUnlockedInternal (
            NoParams gateParams )  [protected]
```

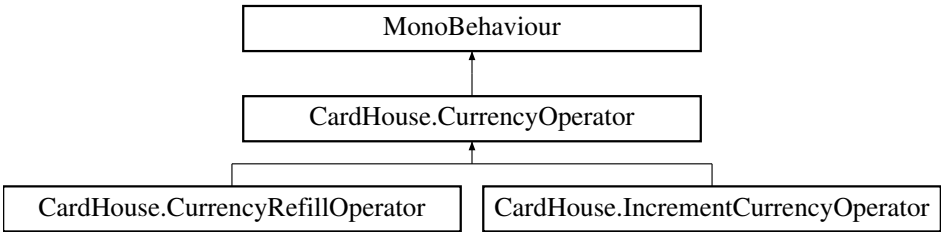
Definition at line 15 of file CurrencyGate.cs.

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

- CurrencyGate.cs

6.29 CardHouse.CurrencyOperator Class Reference

Inheritance diagram for CardHouse.CurrencyOperator:



Public Member Functions

- void Activate ()

Protected Member Functions

- abstract void **AdjustCurrencies** ()

Protected Attributes

- CurrencyRegistry MyRegistry

6.29.1 Detailed Description

Definition at line 5 of file CurrencyOperator.cs.

6.29.2 Member Function Documentation

6.29.2.1 Activate()

```
void CardHouse.CurrencyOperator.Activate ( )
```

Definition at line 18 of file CurrencyOperator.cs.

6.29.3 Member Data Documentation

6.29.3.1 MyRegistry

```
CurrencyRegistry CardHouse.CurrencyOperator.MyRegistry  [protected]
```

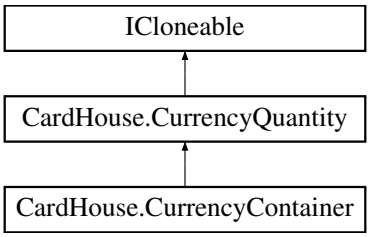
Definition at line 7 of file CurrencyOperator.cs.

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

- CurrencyOperator.cs

6.30 CardHouse.CurrencyQuantity Class Reference

Inheritance diagram for CardHouse.CurrencyQuantity:



Public Member Functions

- virtual object Clone ()

Public Attributes

- CurrencyScriptable CurrencyType
- int Amount

6.30.1 Detailed Description

Definition at line 7 of file CurrencyQuantity.cs.

6.30.2 Member Function Documentation

6.30.2.1 Clone()

```
virtual object CardHouse.CurrencyQuantity.Clone ( )  [virtual]
```

Definition at line 13 of file CurrencyQuantity.cs.

6.30.3 Member Data Documentation

6.30.3.1 Amount

```
int CardHouse.CurrencyQuantity.Amount
```

Definition at line 11 of file CurrencyQuantity.cs.

6.30.3.2 CurrencyType

`CurrencyScriptable CardHouse.CurrencyQuantity.CurrencyType`

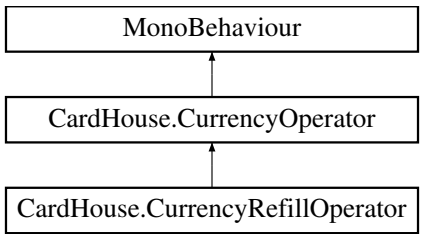
Definition at line 10 of file CurrencyQuantity.cs.

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

- CurrencyQuantity.cs

6.31 CardHouse.CurrencyRefillOperator Class Reference

Inheritance diagram for CardHouse.CurrencyRefillOperator:



Public Attributes

- List< CurrencyScriptable > CurrenciesToRefill

Protected Member Functions

- override void AdjustCurrencies ()
- abstract void **AdjustCurrencies** ()

Additional Inherited Members

Public Member Functions inherited from CardHouse.CurrencyOperator

- void Activate ()

Protected Attributes inherited from CardHouse.CurrencyOperator

- CurrencyRegistry MyRegistry

6.31.1 Detailed Description

Definition at line 6 of file CurrencyRefillOperator.cs.

6.31.2 Member Function Documentation

6.31.2.1 AdjustCurrencies()

```
override void CardHouse.CurrencyRefillOperator.AdjustCurrencies ( )  [protected], [virtual]
```

Implements CardHouse.CurrencyOperator.

Definition at line 11 of file CurrencyRefillOperator.cs.

6.31.3 Member Data Documentation

6.31.3.1 CurrenciesToRefill

```
List<CurrencyScriptable> CardHouse.CurrencyRefillOperator.CurrenciesToRefill
```

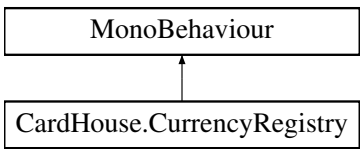
Definition at line 9 of file CurrencyRefillOperator.cs.

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

- CurrencyRefillOperator.cs

6.32 CardHouse.CurrencyRegistry Class Reference

Inheritance diagram for CardHouse.CurrencyRegistry:



Public Member Functions

- int? GetCurrency (string name, int playerIndex)
- int? GetCurrency (CurrencyScriptable resourceDef, int playerIndex)
- void AdjustCurrency (string name, int playerIndex, int amount)
- void Refill (string name, int playerIndex)

Public Attributes

- Text CurrentPlayerLabel
- Transform CurrencyDisplayParent
- GameObject CurrencyDisplayPrefab
- List< CurrencyWallet > PlayerWallets
- Action< int, CurrencyWallet > OnCurrencyChanged

Static Public Attributes

- static CurrencyRegistry Instance

6.32.1 Detailed Description

Definition at line 9 of file CurrencyRegistry.cs.

6.32.2 Member Function Documentation

6.32.2.1 AdjustCurrency()

```
void CardHouse.CurrencyRegistry.AdjustCurrency (
            string name,
            int playerIndex,
            int amount )
```

Definition at line 99 of file CurrencyRegistry.cs.

6.32.2.2 GetCurrency() [1/2]

```
int?  CardHouse.CurrencyRegistry.GetCurrency (
            CurrencyScriptable resourceDef,
            int playerIndex )
```

Definition at line 86 of file CurrencyRegistry.cs.

6.32.2.3 GetCurrency() [2/2]

```
int?  CardHouse.CurrencyRegistry.GetCurrency (
            string name,
            int playerIndex )
```

Definition at line 81 of file CurrencyRegistry.cs.

6.32.2.4 Refill()

```
void CardHouse.CurrencyRegistry.Refill (
            string name,
            int playerIndex )
```

Definition at line 110 of file CurrencyRegistry.cs.

6.32.3 Member Data Documentation

6.32.3.1 CurrencyDisplayParent

```
Transform CardHouse.CurrencyRegistry.CurrencyDisplayParent
```

Definition at line 14 of file CurrencyRegistry.cs.

6.32.3.2 CurrencyDisplayPrefab

```
GameObject CardHouse.CurrencyRegistry.CurrencyDisplayPrefab
```

Definition at line 16 of file CurrencyRegistry.cs.

6.32.3.3 CurrentPlayerLabel

```
Text CardHouse.CurrencyRegistry.CurrentPlayerLabel
```

Definition at line 11 of file CurrencyRegistry.cs.

6.32.3.4 Instance

```
CurrencyRegistry CardHouse.CurrencyRegistry.Instance  [static]
```

Definition at line 21 of file CurrencyRegistry.cs.

6.32.3.5 OnCurrencyChanged

```
Action<int, CurrencyWallet> CardHouse.CurrencyRegistry.OnCurrencyChanged
```

Definition at line 27 of file CurrencyRegistry.cs.

6.32.3.6 PlayerWallets

```
List<CurrencyWallet> CardHouse.CurrencyRegistry.PlayerWallets
```

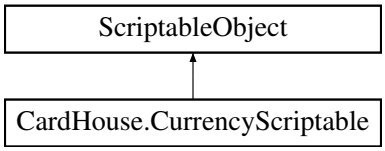
Definition at line 19 of file CurrencyRegistry.cs.

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

- CurrencyRegistry.cs

6.33 CardHouse.CurrencyScriptable Class Reference

Inheritance diagram for CardHouse.CurrencyScriptable:



Public Attributes

- string Name
- Sprite Sprite

6.33.1 Detailed Description

Definition at line 6 of file CurrencyScriptable.cs.

6.33.2 Member Data Documentation

6.33.2.1 Name

```
string CardHouse.CurrencyScriptable.Name
```

Definition at line 8 of file CurrencyScriptable.cs.

6.33.2.2 Sprite

```
Sprite CardHouse.CurrencyScriptable.Sprite
```

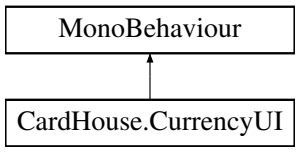
Definition at line 9 of file CurrencyScriptable.cs.

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

- CurrencyScriptable.cs

6.34 CardHouse.CurrencyUI Class Reference

Inheritance diagram for CardHouse.CurrencyUI:



Public Member Functions

- void Apply (CurrencyContainer resource)

Public Attributes

- Image Image
- Text Text

6.34.1 Detailed Description

Definition at line 6 of file CurrencyUI.cs.

6.34.2 Member Function Documentation

6.34.2.1 Apply()

```
void CardHouse.CurrencyUI.Apply (
            CurrencyContainer resource )
```

Definition at line 11 of file CurrencyUI.cs.

6.34.3 Member Data Documentation

6.34.3.1 Image

```
Image CardHouse.CurrencyUI.Image
```

Definition at line 8 of file CurrencyUI.cs.

6.34.3.2 Text

```
Text CardHouse.CurrencyUI.Text
```

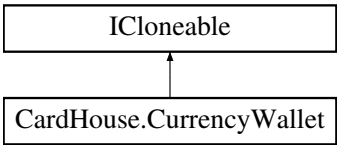
Definition at line 9 of file CurrencyUI.cs.

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

- CurrencyUI.cs

6.35 CardHouse.CurrencyWallet Class Reference

Inheritance diagram for CardHouse.CurrencyWallet:



Public Member Functions

- CurrencyContainer FindCurrency (string name)
- bool CanAfford (List< CurrencyQuantity > Cost)
- object Clone ()

Public Attributes

- List< CurrencyContainer > Currencies

6.35.1 Detailed Description

Definition at line 8 of file CurrencyWallet.cs.

6.35.2 Member Function Documentation

6.35.2.1 CanAfford()

```
bool CardHouse.CurrencyWallet.CanAfford (
            List< CurrencyQuantity > Cost )
```

Definition at line 25 of file CurrencyWallet.cs.

6.35.2.2 Clone()

```
object CardHouse.CurrencyWallet.Clone ( )
```

Definition at line 39 of file CurrencyWallet.cs.

6.35.2.3 FindCurrency()

```
CurrencyContainer CardHouse.CurrencyWallet.FindCurrency (
            string name )
```

Definition at line 12 of file CurrencyWallet.cs.

6.35.3 Member Data Documentation

6.35.3.1 Currencies

```
List<CurrencyContainer> CardHouse.CurrencyWallet.Currencies
```

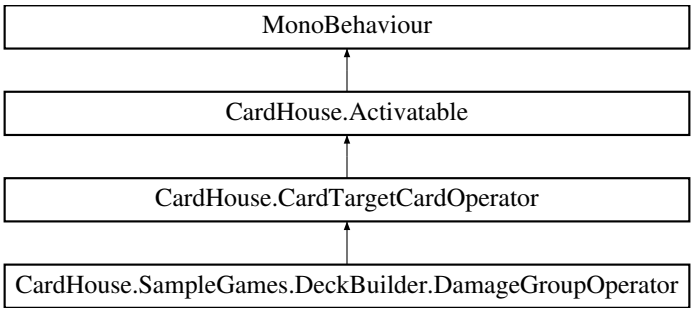
Definition at line 10 of file CurrencyWallet.cs.

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

- CurrencyWallet.cs

6.36 CardHouse.SampleGames.DeckBuilder.DamageGroupOperator Class Reference

Inheritance diagram for CardHouse.SampleGames.DeckBuilder.DamageGroupOperator:



Public Attributes

- int Damage

Public Attributes inherited from CardHouse.CardTargetCardOperator

- SeekerScriptableSet DiscardSeekers

Protected Member Functions

- override void ActOnTarget ()

Protected Member Functions inherited from CardHouse.CardTargetCardOperator

- override void OnActivate ()
- abstract void **ActOnTarget** ()

- virtual void OnActivate ()

Additional Inherited Members

Public Member Functions inherited from CardHouse.Activatable

- void Activate ()

Protected Attributes inherited from CardHouse.CardTargetCardOperator

- Card MyCard
- Card Target

6.36.1 Detailed Description

Definition at line 3 of file DamageGroupOperator.cs.

6.36.2 Member Function Documentation

6.36.2.1 ActOnTarget()

```
override void CardHouse.SampleGames.DeckBuilder.DamageGroupOperator.ActOnTarget ( )  [protected],
[virtual]
```

Implements CardHouse.CardTargetCardOperator.

Definition at line 7 of file DamageGroupOperator.cs.

6.36.3 Member Data Documentation

6.36.3.1 Damage

```
int CardHouse.SampleGames.DeckBuilder.DamageGroupOperator.Damage
```

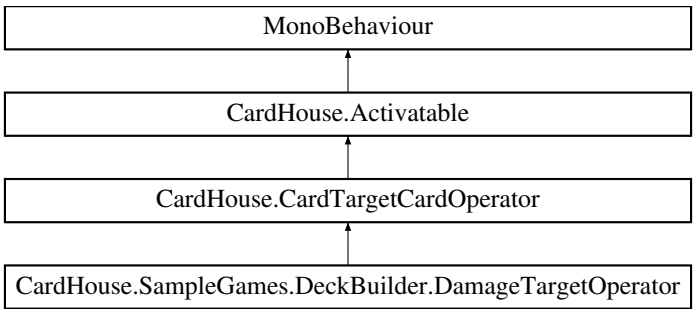
Definition at line 5 of file DamageGroupOperator.cs.

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

- DamageGroupOperator.cs

6.37 CardHouse.SampleGames.DeckBuilder.DamageTargetOperator Class Reference

Inheritance diagram for CardHouse.SampleGames.DeckBuilder.DamageTargetOperator:



Public Attributes

- int Damage

Public Attributes inherited from CardHouse.CardTargetCardOperator

- SeekerScriptableSet DiscardSeekers

Protected Member Functions

- override void ActOnTarget ()

Protected Member Functions inherited from CardHouse.CardTargetCardOperator

- override void OnActivate ()
- abstract void **ActOnTarget** ()

- virtual void OnActivate ()

Additional Inherited Members

Public Member Functions inherited from CardHouse.Activatable

- void Activate ()

Protected Attributes inherited from CardHouse.CardTargetCardOperator

- Card MyCard
- Card Target

6.37.1 Detailed Description

Definition at line 3 of file DamageTargetOperator.cs.

6.37.2 Member Function Documentation

6.37.2.1 ActOnTarget()

```
override void CardHouse.SampleGames.DeckBuilder.DamageTargetOperator.ActOnTarget ( )  [protected],
[virtual]
```

Implements CardHouse.CardTargetCardOperator.

Definition at line 7 of file DamageTargetOperator.cs.

6.37.3 Member Data Documentation

6.37.3.1 Damage

```
int CardHouse.SampleGames.DeckBuilder.DamageTargetOperator.Damage
```

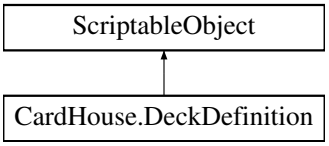
Definition at line 5 of file DamageTargetOperator.cs.

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

- DamageTargetOperator.cs

6.38 CardHouse.DeckDefinition Class Reference

Inheritance diagram for CardHouse.DeckDefinition:



Public Attributes

- Sprite CardBackArt
- List< CardDefinition > CardCollection

6.38.1 Detailed Description

Definition at line 7 of file DeckDefinition.cs.

6.38.2 Member Data Documentation

6.38.2.1 CardBackArt

```
Sprite CardHouse.DeckDefinition.CardBackArt
```

Definition at line 9 of file DeckDefinition.cs.

6.38.2.2 CardCollection

```
List<CardDefinition> CardHouse.DeckDefinition.CardCollection
```

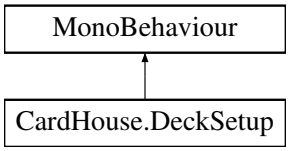
Definition at line 10 of file DeckDefinition.cs.

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

- DeckDefinition.cs

6.39 CardHouse.DeckSetup Class Reference

Inheritance diagram for CardHouse.DeckSetup:



Public Member Functions

- void DoSetup ()

Public Attributes

- bool RunOnStart = true
- GameObject CardPrefab
- DeckDefinition DeckDefinition
- CardGroup Deck
- List< TimedEvent > OnSetupCompleteEventChain

6.39.1 Detailed Description

Definition at line 7 of file DeckSetup.cs.

6.39.2 Member Function Documentation

6.39.2.1 DoSetup()

```
void CardHouse.DeckSetup.DoSetup ( )
```

Definition at line 23 of file DeckSetup.cs.

6.39.3 Member Data Documentation

6.39.3.1 CardPrefab

```
GameObject CardHouse.DeckSetup.CardPrefab
```

Definition at line 10 of file DeckSetup.cs.

6.39.3.2 Deck

```
CardGroup CardHouse.DeckSetup.Deck
```

Definition at line 12 of file DeckSetup.cs.

6.39.3.3 DeckDefinition

```
DeckDefinition CardHouse.DeckSetup.DeckDefinition
```

Definition at line 11 of file DeckSetup.cs.

6.39.3.4 OnSetupCompleteEventChain

```
List<TimedEvent> CardHouse.DeckSetup.OnSetupCompleteEventChain
```

Definition at line 13 of file DeckSetup.cs.

6.39.3.5 RunOnStart

```
bool CardHouse.DeckSetup.RunOnStart = true
```

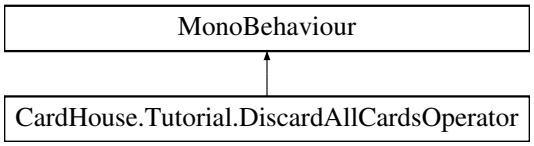
Definition at line 9 of file DeckSetup.cs.

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

- DeckSetup.cs

6.40 CardHouse.Tutorial.DiscardAllCardsOperator Class Reference

Inheritance diagram for CardHouse.Tutorial.DiscardAllCardsOperator:



Public Member Functions

- void Activate ()

Public Attributes

- SeekerScriptableSet DiscardSeekers
- SeekerScriptableSet TargetDiscardSeekers

6.40.1 Detailed Description

Definition at line 6 of file DiscardAllCardsOperator.cs.

6.40.2 Member Function Documentation

6.40.2.1 Activate()

```
void CardHouse.Tutorial.DiscardAllCardsOperator.Activate ( )
```

Definition at line 11 of file DiscardAllCardsOperator.cs.

6.40.3 Member Data Documentation

6.40.3.1 DiscardSeekers

```
SeekerScriptableSet CardHouse.Tutorial.DiscardAllCardsOperator.DiscardSeekers
```

Definition at line 8 of file DiscardAllCardsOperator.cs.

6.40.3.2 TargetDiscardSeekers

```
SeekerScriptableSet CardHouse.Tutorial.DiscardAllCardsOperator.TargetDiscardSeekers
```

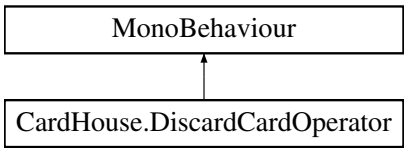
Definition at line 9 of file DiscardAllCardsOperator.cs.

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

- DiscardAllCardsOperator.cs

6.41 CardHouse.DiscardCardOperator Class Reference

Inheritance diagram for CardHouse.DiscardCardOperator:



Public Member Functions

- void Activate ()

6.41.1 Detailed Description

Definition at line 5 of file DiscardCardOperator.cs.

6.41.2 Member Function Documentation

6.41.2.1 Activate()

```
void CardHouse.DiscardCardOperator.Activate ( )
```

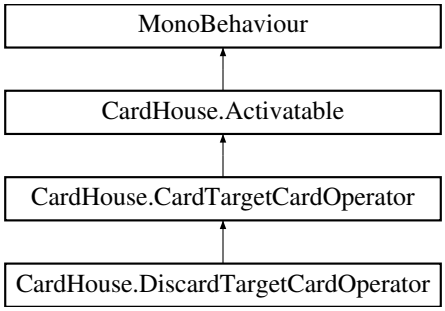
Definition at line 7 of file DiscardCardOperator.cs.

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

- DiscardCardOperator.cs

6.42 CardHouse.DiscardTargetCardOperator Class Reference

Inheritance diagram for CardHouse.DiscardTargetCardOperator:



Public Attributes

- SeekerScriptableSet TargetDiscardSeekers

Public Attributes inherited from CardHouse.CardTargetCardOperator

- SeekerScriptableSet DiscardSeekers

Protected Member Functions

- override void ActOnTarget ()

Protected Member Functions inherited from CardHouse.CardTargetCardOperator

- override void OnActivate ()
- abstract void **ActOnTarget** ()

- virtual void OnActivate ()

Additional Inherited Members

Public Member Functions inherited from CardHouse.Activatable

- void Activate ()

Protected Attributes inherited from CardHouse.CardTargetCardOperator

- Card MyCard
- Card Target

6.42.1 Detailed Description

Definition at line 3 of file DiscardTargetCardOperator.cs.

6.42.2 Member Function Documentation

6.42.2.1 ActOnTarget()

```
override void CardHouse.DiscardTargetCardOperator.ActOnTarget ( )  [protected], [virtual]
```

Implements CardHouse.CardTargetCardOperator.

Definition at line 7 of file DiscardTargetCardOperator.cs.

6.42.3 Member Data Documentation

6.42.3.1 TargetDiscardSeekers

```
SeekerScriptableSet CardHouse.DiscardTargetCardOperator.TargetDiscardSeekers
```

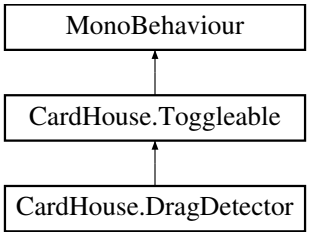
Definition at line 5 of file DiscardTargetCardOperator.cs.

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

- DiscardTargetCardOperator.cs

6.43 CardHouse.DragDetector Class Reference

Inheritance diagram for CardHouse.DragDetector:



Public Attributes

- GateCollection< NoParams > DragGates
- UnityEvent OnDragStart
- GateCollection< DropParams > GroupDropGates
- GateCollection< TargetCardParams > TargetCardGates
- UnityEvent OnDragEnd

Public Attributes inherited from CardHouse.Toggleable

- bool IsActive = true

Additional Inherited Members

Public Member Functions inherited from CardHouse.Toggleable

- void SetIsActive (bool newValue)

6.43.1 Detailed Description

Definition at line 6 of file DragDetector.cs.

6.43.2 Member Data Documentation

6.43.2.1 DragGates

```
GateCollection<NoParams> CardHouse.DragDetector.DragGates
```

Definition at line 8 of file DragDetector.cs.

6.43.2.2 GroupDropGates

```
GateCollection<DropParams> CardHouse.DragDetector.GroupDropGates
```

Definition at line 12 of file DragDetector.cs.

6.43.2.3 OnDragEnd

```
UnityEvent CardHouse.DragDetector.OnDragEnd
```

Definition at line 14 of file DragDetector.cs.

6.43.2.4 OnDragStart

```
UnityEvent CardHouse.DragDetector.OnDragStart
```

Definition at line 9 of file DragDetector.cs.

6.43.2.5 TargetCardGates

```
GateCollection<TargetCardParams> CardHouse.DragDetector.TargetCardGates
```

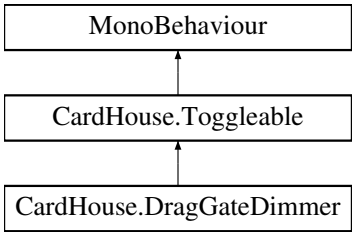
Definition at line 13 of file DragDetector.cs.

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

- DragDetector.cs

6.44 CardHouse.DragGateDimmer Class Reference

Inheritance diagram for CardHouse.DragGateDimmer:



Public Member Functions

- void UpdateHandler ()

Public Member Functions inherited from CardHouse.Toggleable

- void SetIsActive (bool newValue)

Public Attributes

- MultiSpriteOperator Handler
- string ActiveMessage
- string InactiveMessage

Public Attributes inherited from CardHouse.Toggleable

- bool IsActive = true

6.44.1 Detailed Description

Definition at line 6 of file DragGateDimmer.cs.

6.44.2 Member Function Documentation

6.44.2.1 UpdateHandler()

```
void CardHouse.DragGateDimmer.UpdateHandler ( )
```

Definition at line 19 of file DragGateDimmer.cs.

6.44.3 Member Data Documentation

6.44.3.1 ActiveMessage

```
string CardHouse.DragGateDimmer.ActiveMessage
```

Definition at line 9 of file DragGateDimmer.cs.

6.44.3.2 Handler

```
MultiSpriteOperator CardHouse.DragGateDimmer.Handler
```

Definition at line 8 of file DragGateDimmer.cs.

6.44.3.3 InactiveMessage

```
string CardHouse.DragGateDimmer.InactiveMessage
```

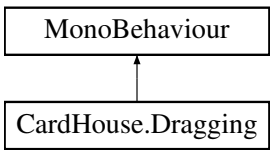
Definition at line 10 of file DragGateDimmer.cs.

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

- DragGateDimmer.cs

6.45 CardHouse.Dragging Class Reference

Inheritance diagram for CardHouse.Dragging:



Public Member Functions

- void UpdateStrategy ()
- Homing GetTarget ()
- void BeginDragging (DragDetector draggable, Homing homing, Turning turning, bool pointUpWhen↩ Dragged=true, float? startingZ=null)
- void StopDragging ()

Public Attributes

- float DefaultCardZ = 0f
- float CardPopupDistance = 1f
- bool UseGrabOffset
- Vector3 GrabOffset
- bool SetNewOffsetOnGrab
- SeekerScriptable< Vector3 > DragHomingStrategy
- Action< DragDetector > OnDrag
- Action< DragDetector > OnDrop
- Action< DragDetector > PostDrop

Static Public Attributes

- static Dragging Instance

6.45.1 Detailed Description

Definition at line 6 of file Dragging.cs.

6.45.2 Member Function Documentation

6.45.2.1 BeginDragging()

```
void CardHouse.Dragging.BeginDragging (
            DragDetector draggable,
            Homing homing,
            Turning turning,
            bool pointUpWhenDragged = true,
            float?  startingZ = null )
```

Definition at line 45 of file Dragging.cs.

6.45.2.2 GetTarget()

```
Homing CardHouse.Dragging.GetTarget ( )
```

Definition at line 40 of file Dragging.cs.

6.45.2.3 StopDragging()

```
void CardHouse.Dragging.StopDragging ( )
```

Definition at line 74 of file Dragging.cs.

6.45.2.4 UpdateStrategy()

```
void CardHouse.Dragging.UpdateStrategy ( )
```

Definition at line 35 of file Dragging.cs.

6.45.3 Member Data Documentation

6.45.3.1 CardPopupDistance

```
float CardHouse.Dragging.CardPopupDistance = 1f
```

Definition at line 9 of file Dragging.cs.

6.45.3.2 DefaultCardZ

```
float CardHouse.Dragging.DefaultCardZ = 0f
```

Definition at line 8 of file Dragging.cs.

6.45.3.3 DragHomingStrategy

```
SeekerScriptable<Vector3> CardHouse.Dragging.DragHomingStrategy
```

Definition at line 15 of file Dragging.cs.

6.45.3.4 GrabOffset

```
Vector3 CardHouse.Dragging.GrabOffset
```

Definition at line 12 of file Dragging.cs.

6.45.3.5 Instance

```
Dragging CardHouse.Dragging.Instance  [static]
```

Definition at line 24 of file Dragging.cs.

6.45.3.6 OnDrag

```
Action<DragDetector> CardHouse.Dragging.OnDrag
```

Definition at line 25 of file Dragging.cs.

6.45.3.7 OnDrop

```
Action<DragDetector> CardHouse.Dragging.OnDrop
```

Definition at line 26 of file Dragging.cs.

6.45.3.8 PostDrop

```
Action<DragDetector> CardHouse.Dragging.PostDrop
```

Definition at line 27 of file Dragging.cs.

6.45.3.9 SetNewOffsetOnGrab

```
bool CardHouse.Dragging.SetNewOffsetOnGrab
```

Definition at line 13 of file Dragging.cs.

6.45.3.10 UseGrabOffset

```
bool CardHouse.Dragging.UseGrabOffset
```

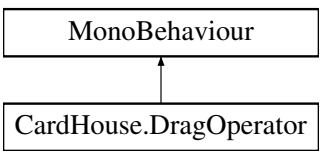
Definition at line 11 of file Dragging.cs.

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

- Dragging.cs

6.46 CardHouse.DragOperator Class Reference

Inheritance diagram for CardHouse.DragOperator:



Public Member Functions

- void SetDragState (bool newState)

Public Attributes

- DragDetector MyDragDetector
- DragAction DragAction
- float DragSwell = 1.2f
- bool PointUpWhenDragged = true
- SeekerScriptableSet PresentationSeekers

6.46.1 Detailed Description

Definition at line 6 of file DragOperator.cs.

6.46.2 Member Function Documentation

6.46.2.1 SetDragState()

```
void CardHouse.DragOperator.SetDragState (
            bool newState )
```

Definition at line 32 of file DragOperator.cs.

6.46.3 Member Data Documentation

6.46.3.1 DragAction

```
DragAction CardHouse.DragOperator.DragAction
```

Definition at line 10 of file DragOperator.cs.

6.46.3.2 DragSwell

```
float CardHouse.DragOperator.DragSwell = 1.2f
```

Definition at line 11 of file DragOperator.cs.

6.46.3.3 MyDragDetector

```
DragDetector CardHouse.DragOperator.MyDragDetector
```

Definition at line 8 of file DragOperator.cs.

6.46.3.4 PointUpWhenDragged

```
bool CardHouse.DragOperator.PointUpWhenDragged = true
```

Definition at line 12 of file DragOperator.cs.

6.46.3.5 PresentationSeekers

```
SeekerScriptableSet CardHouse.DragOperator.PresentationSeekers
```

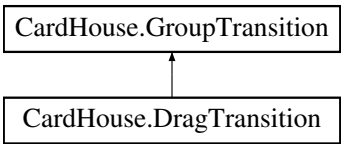
Definition at line 14 of file DragOperator.cs.

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

- DragOperator.cs

6.47 CardHouse.DragTransition Class Reference

Inheritance diagram for CardHouse.DragTransition:



Public Attributes

- DragAction DragAction

Public Attributes inherited from CardHouse.GroupTransition

- CardGroup Source
- CardGroup Destination

6.47.1 Detailed Description

Definition at line 6 of file DragTransition.cs.

6.47.2 Member Data Documentation

6.47.2.1 DragAction

```
DragAction CardHouse.DragTransition.DragAction
```

Definition at line 8 of file DragTransition.cs.

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

- DragTransition.cs

6.48 CardHouse.DropParams Class Reference

Public Attributes

- CardGroup Source
- CardGroup Target
- Card Card
- DragAction DragType

6.48.1 Detailed Description

Definition at line 3 of file DropParams.cs.

6.48.2 Member Data Documentation

6.48.2.1 Card

```
Card CardHouse.DropParams.Card
```

Definition at line 7 of file DropParams.cs.

6.48.2.2 DragType

```
DragAction CardHouse.DropParams.DragType
```

Definition at line 8 of file DropParams.cs.

6.48.2.3 Source

```
CardGroup CardHouse.DropParams.Source
```

Definition at line 5 of file DropParams.cs.

6.48.2.4 Target

```
CardGroup CardHouse.DropParams.Target
```

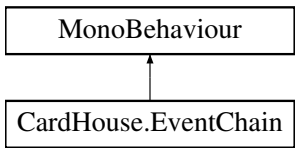
Definition at line 6 of file DropParams.cs.

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

- DropParams.cs

6.49 CardHouse.EventChain Class Reference

Inheritance diagram for CardHouse.EventChain:



Public Member Functions

- void Activate ()

Public Attributes

- List< TimedEvent > Events = new List<TimedEvent>()
- UnityEvent OnChainFinished

6.49.1 Detailed Description

Definition at line 7 of file EventChain.cs.

6.49.2 Member Function Documentation

6.49.2.1 Activate()

```
void CardHouse.EventChain.Activate ( )
```

Definition at line 13 of file EventChain.cs.

6.49.3 Member Data Documentation

6.49.3.1 Events

```
List<TimedEvent> CardHouse.EventChain.Events = new List<TimedEvent>()
```

Definition at line 9 of file EventChain.cs.

6.49.3.2 OnChainFinished

```
UnityEvent CardHouse.EventChain.OnChainFinished
```

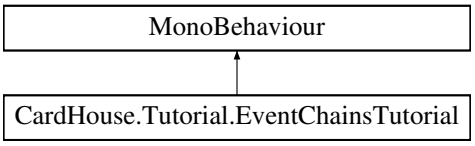
Definition at line 11 of file EventChain.cs.

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

- EventChain.cs

6.50 CardHouse.Tutorial.EventChainsTutorial Class Reference

Inheritance diagram for CardHouse.Tutorial.EventChainsTutorial:



Public Member Functions

- void StartTransition ()

Public Attributes

- EventChain NoChaining
- EventChain Chaining
- EventChain SafeChaining
- TMP_Dropdown Dropdown

6.50.1 Detailed Description

Definition at line 6 of file EventChainsTutorial.cs.

6.50.2 Member Function Documentation

6.50.2.1 StartTransition()

```
void CardHouse.Tutorial.EventChainsTutorial.StartTransition ( )
```

Definition at line 14 of file EventChainsTutorial.cs.

6.50.3 Member Data Documentation

6.50.3.1 Chaining

```
EventChain CardHouse.Tutorial.EventChainsTutorial.Chaining
```

Definition at line 9 of file EventChainsTutorial.cs.

6.50.3.2 Dropdown

```
TMP_Dropdown CardHouse.Tutorial.EventChainsTutorial.Dropdown
```

Definition at line 12 of file EventChainsTutorial.cs.

6.50.3.3 NoChaining

```
EventChain CardHouse.Tutorial.EventChainsTutorial.NoChaining
```

Definition at line 8 of file EventChainsTutorial.cs.

6.50.3.4 SafeChaining

```
EventChain CardHouse.Tutorial.EventChainsTutorial.SafeChaining
```

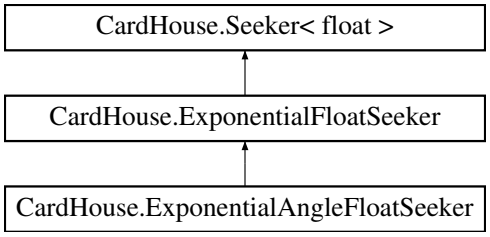
Definition at line 10 of file EventChainsTutorial.cs.

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

- EventChainsTutorial.cs

6.51 CardHouse.ExponentialAngleFloatSeeker Class Reference

Inheritance diagram for CardHouse.ExponentialAngleFloatSeeker:



Public Member Functions

- ExponentialAngleFloatSeeker (float gain=8f, float arrivalDist=0.01f)
- override float Pump (float currentValue, float TimeSinceLastFrame)

Public Member Functions inherited from CardHouse.ExponentialFloatSeeker

- ExponentialFloatSeeker (float gain=8f, float arrivalDist=0.01f)
- override Seeker< float > MakeCopy ()
- override float Pump (float currentValue, float TimeSinceLastFrame)
- override bool IsDone (float currentValue)

Public Member Functions inherited from CardHouse.Seeker< float >

- abstract Seeker< T > **MakeCopy** ()
- void StartSeeking (T from, T to)
- abstract T **Pump** (T currentValue, float TimeSinceLastFrame)
- abstract bool **IsDone** (T currentValue)

Additional Inherited Members

Public Attributes inherited from CardHouse.Seeker< float >

- T End

Protected Attributes inherited from CardHouse.ExponentialFloatSeeker

- float Gain
- float ArrivalDistance

Protected Attributes inherited from CardHouse.Seeker< float >

- T Start

6.51.1 Detailed Description

Definition at line 5 of file ExponentialAngleFloatSeeker.cs.

6.51.2 Constructor & Destructor Documentation

6.51.2.1 ExponentialAngleFloatSeeker()

```
CardHouse.ExponentialAngleFloatSeeker.ExponentialAngleFloatSeeker (
            float gain = 8f,
            float arrivalDist = 0::01f )
```

Definition at line 7 of file ExponentialAngleFloatSeeker.cs.

6.51.3 Member Function Documentation

6.51.3.1 Pump()

```
override float CardHouse.ExponentialAngleFloatSeeker.Pump (
            float currentValue,
            float TimeSinceLastFrame )
```

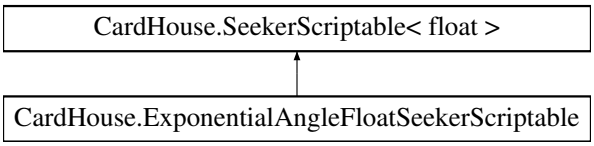
Definition at line 11 of file ExponentialAngleFloatSeeker.cs.

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

- ExponentialAngleFloatSeeker.cs

6.52 CardHouse.ExponentialAngleFloatSeekerScriptable Class Reference

Inheritance diagram for CardHouse.ExponentialAngleFloatSeekerScriptable:



Public Member Functions

- override Seeker< float > GetStrategy (params object[ ] args)
- abstract Seeker< T > **GetStrategy** (params object[ ] args)

Public Attributes

- float Gain = 8f
- float ArrivalDistance = 0.01f

6.52.1 Detailed Description

Definition at line 6 of file ExponentialAngleFloatSeekerScriptable.cs.

6.52.2 Member Function Documentation

6.52.2.1 GetStrategy()

```
override Seeker< float > CardHouse.ExponentialAngleFloatSeekerScriptable.GetStrategy (
            params object[] args )  [virtual]
```

Implements CardHouse.SeekerScriptable< float >.

Definition at line 11 of file ExponentialAngleFloatSeekerScriptable.cs.

6.52.3 Member Data Documentation

6.52.3.1 ArrivalDistance

```
float CardHouse.ExponentialAngleFloatSeekerScriptable.ArrivalDistance = 0.01f
```

Definition at line 9 of file ExponentialAngleFloatSeekerScriptable.cs.

6.52.3.2 Gain

```
float CardHouse.ExponentialAngleFloatSeekerScriptable.Gain = 8f
```

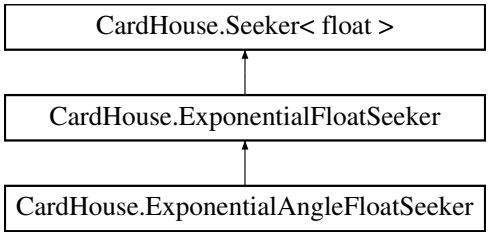
Definition at line 8 of file ExponentialAngleFloatSeekerScriptable.cs.

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

- ExponentialAngleFloatSeekerScriptable.cs

6.53 CardHouse.ExponentialFloatSeeker Class Reference

Inheritance diagram for CardHouse.ExponentialFloatSeeker:



Public Member Functions

- ExponentialFloatSeeker (float gain=8f, float arrivalDist=0.01f)
- override Seeker< float > MakeCopy ()
- override float Pump (float currentValue, float TimeSinceLastFrame)
- override bool IsDone (float currentValue)

Public Member Functions inherited from CardHouse.Seeker< float >

- abstract Seeker< T > **MakeCopy** ()
- void StartSeeking (T from, T to)
- abstract T **Pump** (T currentValue, float TimeSinceLastFrame)
- abstract bool **IsDone** (T currentValue)

Protected Attributes

- float Gain
- float ArrivalDistance

Protected Attributes inherited from CardHouse.Seeker< float >

- T Start

Additional Inherited Members

Public Attributes inherited from CardHouse.Seeker< float >

- T End

6.53.1 Detailed Description

Definition at line 5 of file ExponentialFloatSeeker.cs.

6.53.2 Constructor & Destructor Documentation

6.53.2.1 ExponentialFloatSeeker()

```
CardHouse.ExponentialFloatSeeker.ExponentialFloatSeeker (
            float gain = 8f,
            float arrivalDist = 0::01f )
```

Definition at line 10 of file ExponentialFloatSeeker.cs.

6.53.3 Member Function Documentation

6.53.3.1 IsDone()

```
override bool CardHouse.ExponentialFloatSeeker.IsDone (
            float currentValue )
```

Definition at line 26 of file ExponentialFloatSeeker.cs.

6.53.3.2 MakeCopy()

```
override Seeker< float > CardHouse.ExponentialFloatSeeker.MakeCopy ( )  [virtual]
```

Implements CardHouse.Seeker< float >.

Definition at line 16 of file ExponentialFloatSeeker.cs.

6.53.3.3 Pump()

```
override float CardHouse.ExponentialFloatSeeker.Pump (
            float currentValue,
            float TimeSinceLastFrame )
```

Definition at line 21 of file ExponentialFloatSeeker.cs.

6.53.4 Member Data Documentation

6.53.4.1 ArrivalDistance

```
float CardHouse.ExponentialFloatSeeker.ArrivalDistance  [protected]
```

Definition at line 8 of file ExponentialFloatSeeker.cs.

6.53.4.2 Gain

```
float CardHouse.ExponentialFloatSeeker.Gain  [protected]
```

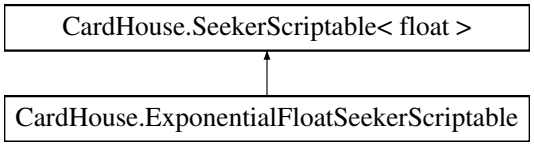
Definition at line 7 of file ExponentialFloatSeeker.cs.

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

- ExponentialFloatSeeker.cs

6.54 CardHouse.ExponentialFloatSeekerScriptable Class Reference

Inheritance diagram for CardHouse.ExponentialFloatSeekerScriptable:



Public Member Functions

- override Seeker< float > GetStrategy (params object[ ] args)
- abstract Seeker< T > **GetStrategy** (params object[ ] args)

Public Attributes

- float Gain = 8f
- float ArrivalDistance = 0.01f

6.54.1 Detailed Description

Definition at line 6 of file ExponentialFloatSeekerScriptable.cs.

6.54.2 Member Function Documentation

6.54.2.1 GetStrategy()

```
override Seeker< float > CardHouse.ExponentialFloatSeekerScriptable.GetStrategy (
            params object[] args )  [virtual]
```

Implements CardHouse.SeekerScriptable< float >.

Definition at line 11 of file ExponentialFloatSeekerScriptable.cs.

6.54.3 Member Data Documentation

6.54.3.1 ArrivalDistance

```
float CardHouse.ExponentialFloatSeekerScriptable.ArrivalDistance = 0.01f
```

Definition at line 9 of file ExponentialFloatSeekerScriptable.cs.

6.54.3.2 Gain

```
float CardHouse.ExponentialFloatSeekerScriptable.Gain = 8f
```

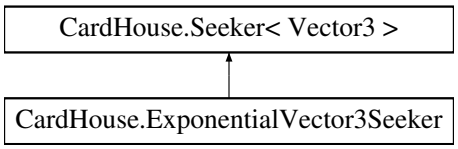
Definition at line 8 of file ExponentialFloatSeekerScriptable.cs.

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

- ExponentialFloatSeekerScriptable.cs

6.55 CardHouse.ExponentialVector3Seeker Class Reference

Inheritance diagram for CardHouse.ExponentialVector3Seeker:



Public Member Functions

- ExponentialVector3Seeker (float xyGain=8f, float zGain=10f, float arrivalDist=0.01f)
- override Seeker< Vector3 > MakeCopy ()
- override Vector3 Pump (Vector3 currentValue, float TimeSinceLastFrame)
- override bool IsDone (Vector3 currentValue)

Public Member Functions inherited from CardHouse.Seeker< Vector3 >

- abstract Seeker< T > **MakeCopy** ()
- void StartSeeking (T from, T to)
- abstract T **Pump** (T currentValue, float TimeSinceLastFrame)
- abstract bool **IsDone** (T currentValue)

Additional Inherited Members

Public Attributes inherited from CardHouse.Seeker< Vector3 >

- T End

Protected Attributes inherited from CardHouse.Seeker< Vector3 >

- T Start

6.55.1 Detailed Description

Definition at line 5 of file ExponentialVector3Seeker.cs.

6.55.2 Constructor & Destructor Documentation

6.55.2.1 ExponentialVector3Seeker()

```
CardHouse.ExponentialVector3Seeker.ExponentialVector3Seeker (
            float xyGain = 8f,
            float zGain = 10f,
            float arrivalDist = 0::01f )
```

Definition at line 11 of file ExponentialVector3Seeker.cs.

6.55.3 Member Function Documentation

6.55.3.1 IsDone()

```
override bool CardHouse.ExponentialVector3Seeker.IsDone (
            Vector3 currentValue )
```

Definition at line 28 of file ExponentialVector3Seeker.cs.

6.55.3.2 MakeCopy()

```
override Seeker< Vector3 > CardHouse.ExponentialVector3Seeker.MakeCopy ( )  [virtual]
```

Implements CardHouse.Seeker< Vector3 >.

Definition at line 18 of file ExponentialVector3Seeker.cs.

6.55.3.3 Pump()

```
override Vector3 CardHouse.ExponentialVector3Seeker.Pump (
            Vector3 currentValue,
            float TimeSinceLastFrame )
```

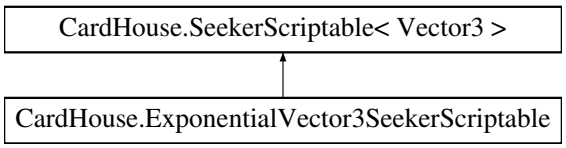
Definition at line 23 of file ExponentialVector3Seeker.cs.

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

- ExponentialVector3Seeker.cs

6.56 CardHouse.ExponentialVector3SeekerScriptable Class Reference

Inheritance diagram for CardHouse.ExponentialVector3SeekerScriptable:



Public Member Functions

- override Seeker< Vector3 > GetStrategy (params object[] args)
- abstract Seeker< T > **GetStrategy** (params object[] args)

Public Attributes

- float XYGain = 8f
- float ZGain = 10f
- float ArrivalDistance = 0.01f

6.56.1 Detailed Description

Definition at line 6 of file ExponentialVector3SeekerScriptable.cs.

6.56.2 Member Function Documentation

6.56.2.1 GetStrategy()

```
override Seeker< Vector3 > CardHouse.ExponentialVector3SeekerScriptable.GetStrategy (
            params object[] args )  [virtual]
```

Implements CardHouse.SeekerScriptable< Vector3 >.

Definition at line 12 of file ExponentialVector3SeekerScriptable.cs.

6.56.3 Member Data Documentation

6.56.3.1 ArrivalDistance

```
float CardHouse.ExponentialVector3SeekerScriptable.ArrivalDistance = 0.01f
```

Definition at line 10 of file ExponentialVector3SeekerScriptable.cs.

6.56.3.2 XYGain

```
float CardHouse.ExponentialVector3SeekerScriptable.XYGain = 8f
```

Definition at line 8 of file ExponentialVector3SeekerScriptable.cs.

6.56.3.3 ZGain

```
float CardHouse.ExponentialVector3SeekerScriptable.ZGain = 10f
```

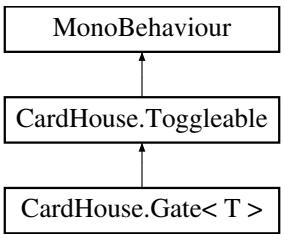
Definition at line 9 of file ExponentialVector3SeekerScriptable.cs.

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

- ExponentialVector3SeekerScriptable.cs

6.57 CardHouse.Gate< T > Class Template Reference

Inheritance diagram for CardHouse.Gate< T >:



Public Member Functions

- bool IsUnlocked (T argObject)

Public Member Functions inherited from CardHouse.Toggleable

- void SetIsActive (bool newValue)

Protected Member Functions

- abstract bool **IsUnlockedInternal** (T argObject)

Additional Inherited Members

Public Attributes inherited from CardHouse.Toggleable

- bool IsActive = true

6.57.1 Detailed Description

Definition at line 3 of file Gate.cs.

6.57.2 Member Function Documentation

6.57.2.1 IsUnlocked()

```
bool CardHouse.Gate< T >.IsUnlocked (
            T argObject )
```

Definition at line 5 of file Gate.cs.

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

- Gate.cs

6.58 CardHouse.GateCollection< T > Class Template Reference

Public Member Functions

- bool AllUnlocked (T gateParams)
- bool AnyUnlocked (T gateParams)

Public Attributes

- List< Gate< T > > Gates

6.58.1 Detailed Description

Definition at line 8 of file GateCollection.cs.

6.58.2 Member Function Documentation

6.58.2.1 AllUnlocked()

```
bool CardHouse.GateCollection< T >.AllUnlocked (
            T gateParams )
```

Definition at line 12 of file GateCollection.cs.

6.58.2.2 AnyUnlocked()

```
bool CardHouse.GateCollection< T >.AnyUnlocked (
            T gateParams )
```

Definition at line 17 of file GateCollection.cs.

6.58.3 Member Data Documentation

6.58.3.1 Gates

```
List<Gate<T> > CardHouse.GateCollection< T >.Gates
```

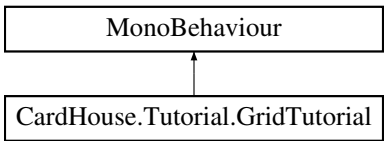
Definition at line 10 of file GateCollection.cs.

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

- GateCollection.cs

6.59 CardHouse.Tutorial.GridTutorial Class Reference

Inheritance diagram for CardHouse.Tutorial.GridTutorial:



Public Member Functions

- void AdjustCardsPerRow ()
- void AdjustCardLimit ()
- void AdjustXScale ()
- void AdjustYScale ()

Public Attributes

- Slider CardsPerRowSlider
- TMP_Text CardsPerRowText
- Slider CardLimitSlider
- TMP_Text CardLimitText
- Slider XScaleSlider
- TMP_Text XScaleText
- Slider YScaleSlider
- TMP_Text YScaleText
- CardGroup Deck
- CardGridLayout Grid

6.59.1 Detailed Description

Definition at line 7 of file GridTutorial.cs.

6.59.2 Member Function Documentation

6.59.2.1 AdjustCardLimit()

```
void CardHouse.Tutorial.GridTutorial.AdjustCardLimit ( )
```

Definition at line 34 of file GridTutorial.cs.

6.59.2.2 AdjustCardsPerRow()

```
void CardHouse.Tutorial.GridTutorial.AdjustCardsPerRow ( )
```

Definition at line 27 of file GridTutorial.cs.

6.59.2.3 AdjustXScale()

```
void CardHouse.Tutorial.GridTutorial.AdjustXScale ( )
```

Definition at line 45 of file GridTutorial.cs.

6.59.2.4 AdjustYScale()

```
void CardHouse.Tutorial.GridTutorial.AdjustYScale ( )
```

Definition at line 52 of file GridTutorial.cs.

6.59.3 Member Data Documentation

6.59.3.1 CardLimitSlider

```
Slider CardHouse.Tutorial.GridTutorial.CardLimitSlider
```

Definition at line 11 of file GridTutorial.cs.

6.59.3.2 CardLimitText

```
TMP_Text CardHouse.Tutorial.GridTutorial.CardLimitText
```

Definition at line 12 of file GridTutorial.cs.

6.59.3.3 CardsPerRowSlider

```
Slider CardHouse.Tutorial.GridTutorial.CardsPerRowSlider
```

Definition at line 9 of file GridTutorial.cs.

6.59.3.4 CardsPerRowText

```
TMP_Text CardHouse.Tutorial.GridTutorial.CardsPerRowText
```

Definition at line 10 of file GridTutorial.cs.

6.59.3.5 Deck

```
CardGroup CardHouse.Tutorial.GridTutorial.Deck
```

Definition at line 18 of file GridTutorial.cs.

6.59.3.6 Grid

```
CardGridLayout CardHouse.Tutorial.GridTutorial.Grid
```

Definition at line 19 of file GridTutorial.cs.

6.59.3.7 XScaleSlider

```
Slider CardHouse.Tutorial.GridTutorial.XScaleSlider
```

Definition at line 13 of file GridTutorial.cs.

6.59.3.8 XScaleText

```
TMP_Text CardHouse.Tutorial.GridTutorial.XScaleText
```

Definition at line 14 of file GridTutorial.cs.

6.59.3.9 YScaleSlider

```
Slider CardHouse.Tutorial.GridTutorial.YScaleSlider
```

Definition at line 15 of file GridTutorial.cs.

6.59.3.10 YScaleText

```
TMP_Text CardHouse.Tutorial.GridTutorial.YScaleText
```

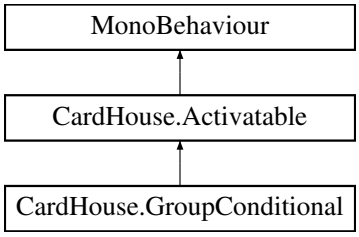
Definition at line 16 of file GridTutorial.cs.

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

- GridTutorial.cs

6.60 CardHouse.GroupConditional Class Reference

Inheritance diagram for CardHouse.GroupConditional:



Public Attributes

- Card MyCard
- List< GroupNameUnityActionKvp > Responses

Protected Member Functions

- override void OnActivate ()

- virtual void OnActivate ()

Additional Inherited Members

Public Member Functions inherited from CardHouse.Activatable

- void Activate ()

6.60.1 Detailed Description

Definition at line 7 of file GroupConditional.cs.

6.60.2 Member Function Documentation

6.60.2.1 OnActivate()

```
override void CardHouse.GroupConditional.OnActivate ( )  [protected], [virtual]
```

Reimplemented from CardHouse.Activatable.

Definition at line 12 of file GroupConditional.cs.

6.60.3 Member Data Documentation

6.60.3.1 MyCard

```
Card CardHouse.GroupConditional.MyCard
```

Definition at line 9 of file GroupConditional.cs.

6.60.3.2 Responses

```
List<GroupNameUnityActionKvp> CardHouse.GroupConditional.Responses
```

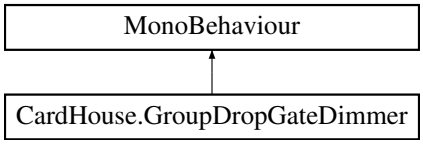
Definition at line 10 of file GroupConditional.cs.

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

- GroupConditional.cs

6.61 CardHouse.GroupDropGateDimmer Class Reference

Inheritance diagram for CardHouse.GroupDropGateDimmer:



Public Attributes

- SpriteOperator Handler
- string ActiveMessage
- string InactiveMessage

6.61.1 Detailed Description

Definition at line 6 of file GroupDropGateDimmer.cs.

6.61.2 Member Data Documentation

6.61.2.1 ActiveMessage

```
string CardHouse.GroupDropGateDimmer.ActiveMessage
```

Definition at line 9 of file GroupDropGateDimmer.cs.

6.61.2.2 Handler

```
SpriteOperator CardHouse.GroupDropGateDimmer.Handler
```

Definition at line 8 of file GroupDropGateDimmer.cs.

6.61.2.3 InactiveMessage

```
string CardHouse.GroupDropGateDimmer.InactiveMessage
```

Definition at line 10 of file GroupDropGateDimmer.cs.

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

- GroupDropGateDimmer.cs

6.62 CardHouse.GroupNameUnityActionKvp Class Reference

Public Attributes

- GroupName Key
- UnityEvent Value

6.62.1 Detailed Description

Definition at line 26 of file GroupConditional.cs.

6.62.2 Member Data Documentation

6.62.2.1 Key

```
GroupName CardHouse.GroupNameUnityActionKvp.Key
```

Definition at line 28 of file GroupConditional.cs.

6.62.2.2 Value

```
UnityEvent CardHouse.GroupNameUnityActionKvp.Value
```

Definition at line 29 of file GroupConditional.cs.

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

- GroupConditional.cs

6.63 CardHouse.GroupSetup.GroupPopulationData Struct Reference

Public Attributes

- CardGroup Group
- GameObject CardPrefab
- int CardCount

6.63.1 Detailed Description

Definition at line 11 of file GroupSetup.cs.

6.63.2 Member Data Documentation

6.63.2.1 CardCount

```
int CardHouse.GroupSetup.GroupPopulationData.CardCount
```

Definition at line 15 of file GroupSetup.cs.

6.63.2.2 CardPrefab

```
GameObject CardHouse.GroupSetup.GroupPopulationData.CardPrefab
```

Definition at line 14 of file GroupSetup.cs.

6.63.2.3 Group

```
CardGroup CardHouse.GroupSetup.GroupPopulationData.Group
```

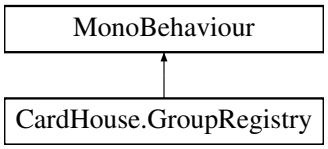
Definition at line 13 of file GroupSetup.cs.

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

- GroupSetup.cs

6.64 CardHouse.GroupRegistry Class Reference

Inheritance diagram for CardHouse.GroupRegistry:



Classes

- class NamedGroup

Public Member Functions

- CardGroup Get (GroupName name, int? playerIndex)
- GroupName GetGroupName (CardGroup group)
- Loyalty GetLoyalty (CardGroup group, int playerIndex)
- int? GetOwnerIndex (CardGroup group)

Public Attributes

- List< NamedGroup > Groups = new List<NamedGroup>()

Static Public Attributes

- static GroupRegistry Instance

6.64.1 Detailed Description

Definition at line 7 of file GroupRegistry.cs.

6.64.2 Member Function Documentation

6.64.2.1 Get()

```
CardGroup CardHouse.GroupRegistry.Get (
            GroupName name,
            int?  playerIndex )
```

Definition at line 26 of file GroupRegistry.cs.

6.64.2.2 GetGroupName()

```
GroupName CardHouse.GroupRegistry.GetGroupName (
            CardGroup group )
```

Definition at line 38 of file GroupRegistry.cs.

6.64.2.3 GetLoyalty()

```
Loyalty CardHouse.GroupRegistry.GetLoyalty (
            CardGroup group,
            int playerIndex )
```

Definition at line 50 of file GroupRegistry.cs.

6.64.2.4 GetOwnerIndex()

```
int?  CardHouse.GroupRegistry.GetOwnerIndex (
            CardGroup group )
```

Definition at line 59 of file GroupRegistry.cs.

6.64.3 Member Data Documentation

6.64.3.1 Groups

```
List<NamedGroup> CardHouse.GroupRegistry.Groups = new List<NamedGroup>()
```

Definition at line 17 of file GroupRegistry.cs.

6.64.3.2 Instance

```
GroupRegistry CardHouse.GroupRegistry.Instance  [static]
```

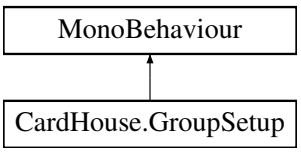
Definition at line 19 of file GroupRegistry.cs.

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

- GroupRegistry.cs

6.65 CardHouse.GroupSetup Class Reference

Inheritance diagram for CardHouse.GroupSetup:



Classes

- struct GroupPopulationData

Public Member Functions

- void DoSetup ()

Public Attributes

- bool RunOnStart = true
- List< GroupPopulationData > GroupPopulationList
- List< CardGroup > GroupsToShuffle
- List< TimedEvent > OnSetupCompleteEventChain

6.65.1 Detailed Description

Definition at line 8 of file GroupSetup.cs.

6.65.2 Member Function Documentation

6.65.2.1 DoSetup()

```
void CardHouse.GroupSetup.DoSetup ( )
```

Definition at line 34 of file GroupSetup.cs.

6.65.3 Member Data Documentation

6.65.3.1 GroupPopulationList

```
List<GroupPopulationData> CardHouse.GroupSetup.GroupPopulationList
```

Definition at line 20 of file GroupSetup.cs.

6.65.3.2 GroupsToShuffle

```
List<CardGroup> CardHouse.GroupSetup.GroupsToShuffle
```

Definition at line 22 of file GroupSetup.cs.

6.65.3.3 OnSetupCompleteEventChain

```
List<TimedEvent> CardHouse.GroupSetup.OnSetupCompleteEventChain
```

Definition at line 24 of file GroupSetup.cs.

6.65.3.4 RunOnStart

```
bool CardHouse.GroupSetup.RunOnStart = true
```

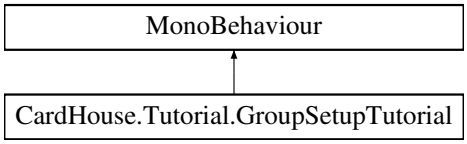
Definition at line 18 of file GroupSetup.cs.

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

- GroupSetup.cs

6.66 CardHouse.Tutorial.GroupSetupTutorial Class Reference

Inheritance diagram for CardHouse.Tutorial.GroupSetupTutorial:



Public Member Functions

- void Setup ()
- void AdjustShuffle ()
- void AdjustASpadesSlider ()
- void AdjustQDiamondsSlider ()
- void AdjustHearts10Slider ()

Public Attributes

- GroupSetup SetupComponent
- CardTransferOperator PullBackOperator
- CardGroup Deck
- TMP_Text ASpadesText
- Slider ASpadesSlider
- TMP_Text QDiamondsText
- Slider QDiamondsSlider
- TMP_Text Hearts10Text
- Slider Hearts10Slider
- Toggle ShuffleToggle

6.66.1 Detailed Description

Definition at line 9 of file GroupSetupTutorial.cs.

6.66.2 Member Function Documentation

6.66.2.1 AdjustASpadesSlider()

```
void CardHouse.Tutorial.GroupSetupTutorial.AdjustASpadesSlider ( )
```

Definition at line 44 of file GroupSetupTutorial.cs.

6.66.2.2 AdjustHearts10Slider()

```
void CardHouse.Tutorial.GroupSetupTutorial.AdjustHearts10Slider ( )
```

Definition at line 60 of file GroupSetupTutorial.cs.

6.66.2.3 AdjustQDiamondsSlider()

```
void CardHouse.Tutorial.GroupSetupTutorial.AdjustQDiamondsSlider ( )
```

Definition at line 52 of file GroupSetupTutorial.cs.

6.66.2.4 AdjustShuffle()

```
void CardHouse.Tutorial.GroupSetupTutorial.AdjustShuffle ( )
```

Definition at line 35 of file GroupSetupTutorial.cs.

6.66.2.5 Setup()

```
void CardHouse.Tutorial.GroupSetupTutorial.Setup ( )
```

Definition at line 23 of file GroupSetupTutorial.cs.

6.66.3 Member Data Documentation

6.66.3.1 ASpadesSlider

```
Slider CardHouse.Tutorial.GroupSetupTutorial.ASpadesSlider
```

Definition at line 16 of file GroupSetupTutorial.cs.

6.66.3.2 ASpadesText

```
TMP_Text CardHouse.Tutorial.GroupSetupTutorial.ASpadesText
```

Definition at line 15 of file GroupSetupTutorial.cs.

6.66.3.3 Deck

```
CardGroup CardHouse.Tutorial.GroupSetupTutorial.Deck
```

Definition at line 13 of file GroupSetupTutorial.cs.

6.66.3.4 Hearts10Slider

```
Slider CardHouse.Tutorial.GroupSetupTutorial.Hearts10Slider
```

Definition at line 20 of file GroupSetupTutorial.cs.

6.66.3.5 Hearts10Text

```
TMP_Text CardHouse.Tutorial.GroupSetupTutorial.Hearts10Text
```

Definition at line 19 of file GroupSetupTutorial.cs.

6.66.3.6 PullBackOperator

```
CardTransferOperator CardHouse.Tutorial.GroupSetupTutorial.PullBackOperator
```

Definition at line 12 of file GroupSetupTutorial.cs.

6.66.3.7 QDiamondsSlider

```
Slider CardHouse.Tutorial.GroupSetupTutorial.QDiamondsSlider
```

Definition at line 18 of file GroupSetupTutorial.cs.

6.66.3.8 QDiamondsText

```
TMP_Text CardHouse.Tutorial.GroupSetupTutorial.QDiamondsText
```

Definition at line 17 of file GroupSetupTutorial.cs.

6.66.3.9 SetupComponent

```
GroupSetup CardHouse.Tutorial.GroupSetupTutorial.SetupComponent
```

Definition at line 11 of file GroupSetupTutorial.cs.

6.66.3.10 ShuffleToggle

```
Toggle CardHouse.Tutorial.GroupSetupTutorial.ShuffleToggle
```

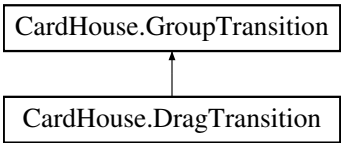
Definition at line 21 of file GroupSetupTutorial.cs.

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

- GroupSetupTutorial.cs

6.67 CardHouse.GroupTransition Class Reference

Inheritance diagram for CardHouse.GroupTransition:



Public Attributes

- CardGroup Source
- CardGroup Destination

6.67.1 Detailed Description

Definition at line 6 of file GroupTransition.cs.

6.67.2 Member Data Documentation

6.67.2.1 Destination

```
CardGroup CardHouse.GroupTransition.Destination
```

Definition at line 9 of file GroupTransition.cs.

6.67.2.2 Source

```
CardGroup CardHouse.GroupTransition.Source
```

Definition at line 8 of file GroupTransition.cs.

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

- GroupTransition.cs

6.68 CardHouse.MultiplayerBoardSetup.GroupTransitionByName Class Reference

Public Attributes

- int PhaseIndex
- GroupName Source
- GroupName Destination
- DragAction DragAction
- PvpMode Mode

6.68.1 Detailed Description

Definition at line 13 of file MultiplayerBoardSetup.cs.

6.68.2 Member Data Documentation

6.68.2.1 Destination

```
GroupName CardHouse.MultiplayerBoardSetup.GroupTransitionByName.Destination
```

Definition at line 17 of file MultiplayerBoardSetup.cs.

6.68.2.2 DragAction

```
DragAction CardHouse.MultiplayerBoardSetup.GroupTransitionByName.DragAction
```

Definition at line 18 of file MultiplayerBoardSetup.cs.

6.68.2.3 Mode

```
PvpMode CardHouse.MultiplayerBoardSetup.GroupTransitionByName.Mode
```

Definition at line 19 of file MultiplayerBoardSetup.cs.

6.68.2.4 PhaseIndex

```
int CardHouse.MultiplayerBoardSetup.GroupTransitionByName.PhaseIndex
```

Definition at line 15 of file MultiplayerBoardSetup.cs.

6.68.2.5 Source

```
GroupName CardHouse.MultiplayerBoardSetup.GroupTransitionByName.Source
```

Definition at line 16 of file MultiplayerBoardSetup.cs.

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

- MultiplayerBoardSetup.cs

6.69 CardHouse.Card.GroupTransitionEvent Class Reference

Public Attributes

- GroupName Group
- UnityEvent EntryEvent
- UnityEvent ExitEvent

6.69.1 Detailed Description

Definition at line 12 of file Card.cs.

6.69.2 Member Data Documentation

6.69.2.1 EntryEvent

```
UnityEvent CardHouse.Card.GroupTransitionEvent.EntryEvent
```

Definition at line 15 of file Card.cs.

6.69.2.2 ExitEvent

```
UnityEvent CardHouse.Card.GroupTransitionEvent.ExitEvent
```

Definition at line 16 of file Card.cs.

6.69.2.3 Group

```
GroupName CardHouse.Card.GroupTransitionEvent.Group
```

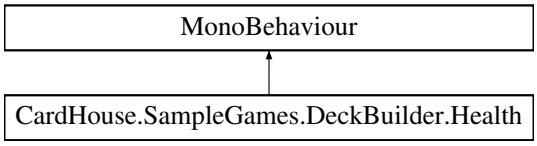
Definition at line 14 of file Card.cs.

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

- Card.cs

6.70 CardHouse.SampleGames.DeckBuilder.Health Class Reference

Inheritance diagram for CardHouse.SampleGames.DeckBuilder.Health:



Public Member Functions

- void Change (int diff)

Public Attributes

- TextMeshPro HealthText
- int HealthLevel
- UnityEvent OnDeath

6.70.1 Detailed Description

Definition at line 7 of file Health.cs.

6.70.2 Member Function Documentation

6.70.2.1 Change()

```
void CardHouse.SampleGames.DeckBuilder.Health.Change (
            int diff )
```

Definition at line 23 of file Health.cs.

6.70.3 Member Data Documentation

6.70.3.1 HealthLevel

```
int CardHouse.SampleGames.DeckBuilder.Health.HealthLevel
```

Definition at line 10 of file Health.cs.

6.70.3.2 HealthText

```
TextMeshPro CardHouse.SampleGames.DeckBuilder.Health.HealthText
```

Definition at line 9 of file Health.cs.

6.70.3.3 OnDeath

```
UnityEvent CardHouse.SampleGames.DeckBuilder.Health.OnDeath
```

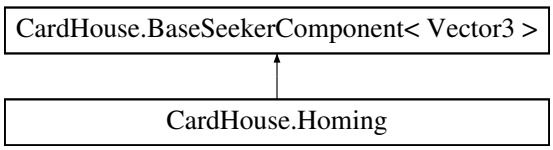
Definition at line 11 of file Health.cs.

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

- Health.cs

6.71 CardHouse.Homing Class Reference

Inheritance diagram for CardHouse.Homing:



Protected Member Functions

- override Seeker< Vector3 > GetDefaultSeeker ()
- override Vector3 GetCurrentValue ()
- override void SetNewValue (Vector3 value)

Protected Member Functions inherited from CardHouse.BaseSeekerComponent< Vector3 >

- abstract Seeker< T > **GetDefaultSeeker** ()
- abstract T **GetCurrentValue** ()
- abstract void **SetNewValue** (T value)

Additional Inherited Members

Public Member Functions inherited from CardHouse.BaseSeekerComponent< Vector3 >

- void StartSeeking (T destination, Seeker< T > strategy=null, bool useLocalSpace=false)

Public Attributes inherited from CardHouse.BaseSeekerComponent< Vector3 >

- SeekerScriptable< T > Strategy

Protected Attributes inherited from CardHouse.BaseSeekerComponent< Vector3 >

- Seeker< T > MyStrategy
- bool IsSeeking
- bool UseLocalSpace

6.71.1 Detailed Description

Definition at line 5 of file Homing.cs.

6.71.2 Member Function Documentation

6.71.2.1 GetCurrentValue()

```
override Vector3 CardHouse.Homing.GetCurrentValue ( )  [protected], [virtual]
```

Implements CardHouse.BaseSeekerComponent< Vector3 >.

Definition at line 12 of file Homing.cs.

6.71.2.2 GetDefaultSeeker()

```
override Seeker< Vector3 > CardHouse.Homing.GetDefaultSeeker ( )  [protected], [virtual]
```

Implements CardHouse.BaseSeekerComponent< Vector3 >.

Definition at line 7 of file Homing.cs.

6.71.2.3 SetNewValue()

```
override void CardHouse.Homing.SetNewValue (
            Vector3 value )  [protected]
```

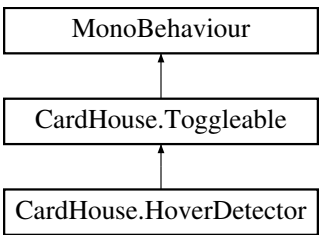
Definition at line 17 of file Homing.cs.

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

- Homing.cs

6.72 CardHouse.HoverDetector Class Reference

Inheritance diagram for CardHouse.HoverDetector:



Public Attributes

- UnityEvent OnHover
- UnityEvent OnUnHover

Public Attributes inherited from CardHouse.Toggleable

- bool IsActive = true

Additional Inherited Members

Public Member Functions inherited from CardHouse.Toggleable

- void SetIsActive (bool newValue)

6.72.1 Detailed Description

Definition at line 5 of file HoverDetector.cs.

6.72.2 Member Data Documentation

6.72.2.1 OnHover

```
UnityEvent CardHouse.HoverDetector.OnHover
```

Definition at line 7 of file HoverDetector.cs.

6.72.2.2 OnUnHover

```
UnityEvent CardHouse.HoverDetector.OnUnHover
```

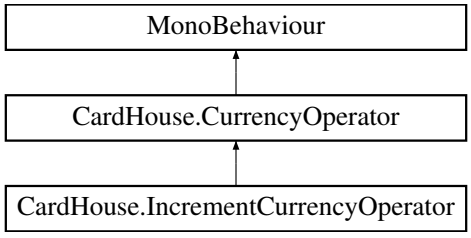
Definition at line 8 of file HoverDetector.cs.

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

- HoverDetector.cs

6.73 CardHouse.IncrementCurrencyOperator Class Reference

Inheritance diagram for CardHouse.IncrementCurrencyOperator:



Public Attributes

- List< CurrencyQuantity > CurrenciesToChange

Protected Member Functions

- override void AdjustCurrencies ()
- abstract void **AdjustCurrencies** ()

Additional Inherited Members

Public Member Functions inherited from CardHouse.CurrencyOperator

- void Activate ()

Protected Attributes inherited from CardHouse.CurrencyOperator

- CurrencyRegistry MyRegistry

6.73.1 Detailed Description

Definition at line 6 of file IncrementCurrencyOperator.cs.

6.73.2 Member Function Documentation

6.73.2.1 AdjustCurrencies()

```
override void CardHouse.IncrementCurrencyOperator.AdjustCurrencies ( )  [protected], [virtual]
```

Implements CardHouse.CurrencyOperator.

Definition at line 11 of file IncrementCurrencyOperator.cs.

6.73.3 Member Data Documentation

6.73.3.1 CurrenciesToChange

```
List<CurrencyQuantity> CardHouse.IncrementCurrencyOperator.CurrenciesToChange
```

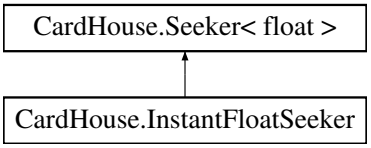
Definition at line 9 of file IncrementCurrencyOperator.cs.

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

- IncrementCurrencyOperator.cs

6.74 CardHouse.InstantFloatSeeker Class Reference

Inheritance diagram for CardHouse.InstantFloatSeeker:



Public Member Functions

- override Seeker< float > MakeCopy ()
- override float Pump (float currentValue, float TimeSinceLastFrame)
- override bool IsDone (float currentValue)

Public Member Functions inherited from CardHouse.Seeker< float >

- abstract Seeker< T > **MakeCopy** ()
- void StartSeeking (T from, T to)
- abstract T **Pump** (T currentValue, float TimeSinceLastFrame)
- abstract bool **IsDone** (T currentValue)

Additional Inherited Members

Public Attributes inherited from CardHouse.Seeker< float >

- T End

Protected Attributes inherited from CardHouse.Seeker< float >

- T Start

6.74.1 Detailed Description

Definition at line 3 of file InstantFloatSeeker.cs.

6.74.2 Member Function Documentation

6.74.2.1 IsDone()

```
override bool CardHouse.InstantFloatSeeker.IsDone (
            float currentValue )
```

Definition at line 15 of file InstantFloatSeeker.cs.

6.74.2.2 MakeCopy()

```
override Seeker< float > CardHouse.InstantFloatSeeker.MakeCopy ( )  [virtual]
```

Implements CardHouse.Seeker< float >.

Definition at line 5 of file InstantFloatSeeker.cs.

6.74.2.3 Pump()

```
override float CardHouse.InstantFloatSeeker.Pump (
            float currentValue,
            float TimeSinceLastFrame )
```

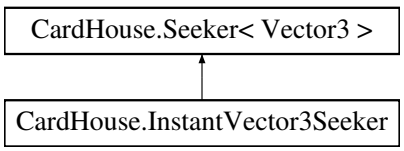
Definition at line 10 of file InstantFloatSeeker.cs.

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

- InstantFloatSeeker.cs

6.75 CardHouse.InstantVector3Seeker Class Reference

Inheritance diagram for CardHouse.InstantVector3Seeker:



Public Member Functions

- override Seeker< Vector3 > MakeCopy ()
- override Vector3 Pump (Vector3 currentValue, float TimeSinceLastFrame)
- override bool IsDone (Vector3 currentValue)

Public Member Functions inherited from CardHouse.Seeker< Vector3 >

- abstract Seeker< T > **MakeCopy** ()
- void StartSeeking (T from, T to)
- abstract T **Pump** (T currentValue, float TimeSinceLastFrame)
- abstract bool **IsDone** (T currentValue)

Additional Inherited Members

Public Attributes inherited from CardHouse.Seeker< Vector3 >

- T End

Protected Attributes inherited from CardHouse.Seeker< Vector3 >

- T Start

6.75.1 Detailed Description

Definition at line 5 of file InstantVector3Seeker.cs.

6.75.2 Member Function Documentation

6.75.2.1 IsDone()

```
override bool CardHouse.InstantVector3Seeker.IsDone (
            Vector3 currentValue )
```

Definition at line 17 of file InstantVector3Seeker.cs.

6.75.2.2 MakeCopy()

```
override Seeker< Vector3 > CardHouse.InstantVector3Seeker.MakeCopy ( )  [virtual]
```

Implements CardHouse.Seeker< Vector3 >.

Definition at line 7 of file InstantVector3Seeker.cs.

6.75.2.3 Pump()

```
override Vector3 CardHouse.InstantVector3Seeker.Pump (
            Vector3 currentValue,
            float TimeSinceLastFrame )
```

Definition at line 12 of file InstantVector3Seeker.cs.

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

- InstantVector3Seeker.cs

6.76 CardHouse.Tutorial.MultiBoardTutorial.InstructionImagePair Class Reference

Public Attributes

- Sprite Image
- string Text

6.76.1 Detailed Description

Definition at line 12 of file MultiBoardTutorial.cs.

6.76.2 Member Data Documentation

6.76.2.1 Image

```
Sprite CardHouse.Tutorial.MultiBoardTutorial.InstructionImagePair.Image
```

Definition at line 14 of file MultiBoardTutorial.cs.

6.76.2.2 Text

```
string CardHouse.Tutorial.MultiBoardTutorial.InstructionImagePair.Text
```

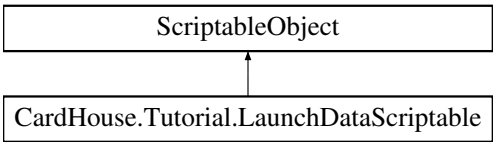
Definition at line 16 of file MultiBoardTutorial.cs.

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

- MultiBoardTutorial.cs

6.77 CardHouse.Tutorial.LaunchDataScriptable Class Reference

Inheritance diagram for CardHouse.Tutorial.LaunchDataScriptable:



Public Attributes

- bool LaunchedTutorial
- string ActiveScene
- List< string > OpenScenes

6.77.1 Detailed Description

Definition at line 6 of file LaunchDataScriptable.cs.

6.77.2 Member Data Documentation

6.77.2.1 ActiveScene

```
string CardHouse.Tutorial.LaunchDataScriptable.ActiveScene
```

Definition at line 9 of file LaunchDataScriptable.cs.

6.77.2.2 LaunchedTutorial

```
bool CardHouse.Tutorial.LaunchDataScriptable.LaunchedTutorial
```

Definition at line 8 of file LaunchDataScriptable.cs.

6.77.2.3 OpenScenes

```
List<string> CardHouse.Tutorial.LaunchDataScriptable.OpenScenes
```

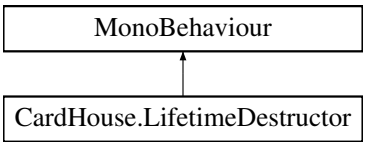
Definition at line 10 of file LaunchDataScriptable.cs.

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

- LaunchDataScriptable.cs

6.78 CardHouse.LifetimeDestructor Class Reference

Inheritance diagram for CardHouse.LifetimeDestructor:



Public Attributes

- float Lifetime = 1f

6.78.1 Detailed Description

Definition at line 6 of file LifetimeDestructor.cs.

6.78.2 Member Data Documentation

6.78.2.1 Lifetime

```
float CardHouse.LifetimeDestructor.Lifetime = 1f
```

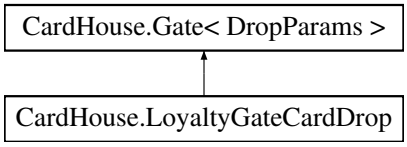
Definition at line 8 of file LifetimeDestructor.cs.

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

- LifetimeDestructor.cs

6.79 CardHouse.LoyaltyGateCardDrop Class Reference

Inheritance diagram for CardHouse.LoyaltyGateCardDrop:



Public Attributes

- Loyalty Loyalty
- List< GroupName > Destinations

Protected Member Functions

- override bool IsUnlockedInternal (DropParams gateParams)

Protected Member Functions inherited from CardHouse.Gate< DropParams >

- abstract bool **IsUnlockedInternal** (T argObject)

Additional Inherited Members

Public Member Functions inherited from CardHouse.Gate< DropParams >

- bool IsUnlocked (T argObject)

6.79.1 Detailed Description

Definition at line 7 of file LoyaltyGateCardDrop.cs.

6.79.2 Member Function Documentation

6.79.2.1 IsUnlockedInternal()

```
override bool CardHouse.LoyaltyGateCardDrop.IsUnlockedInternal (
            DropParams gateParams )  [protected]
```

Definition at line 16 of file LoyaltyGateCardDrop.cs.

6.79.3 Member Data Documentation

6.79.3.1 Destinations

```
List<GroupName> CardHouse.LoyaltyGateCardDrop.Destinations
```

Definition at line 10 of file LoyaltyGateCardDrop.cs.

6.79.3.2 Loyalty

```
Loyalty CardHouse.LoyaltyGateCardDrop.Loyalty
```

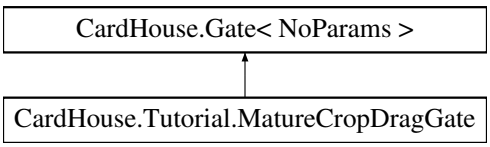
Definition at line 9 of file LoyaltyGateCardDrop.cs.

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

- LoyaltyGateCardDrop.cs

6.80 CardHouse.Tutorial.MatureCropDragGate Class Reference

Inheritance diagram for CardHouse.Tutorial.MatureCropDragGate:



Protected Member Functions

- override bool IsUnlockedInternal (NoParams gateParams)

Protected Member Functions inherited from CardHouse.Gate< NoParams >

- abstract bool **IsUnlockedInternal** (T argObject)

Additional Inherited Members

Public Member Functions inherited from CardHouse.Gate< NoParams >

- bool IsUnlocked (T argObject)

6.80.1 Detailed Description

Definition at line 3 of file MatureCropDragGate.cs.

6.80.2 Member Function Documentation

6.80.2.1 IsUnlockedInternal()

```
override bool CardHouse.Tutorial.MatureCropDragGate.IsUnlockedInternal (
            NoParams gateParams )  [protected]
```

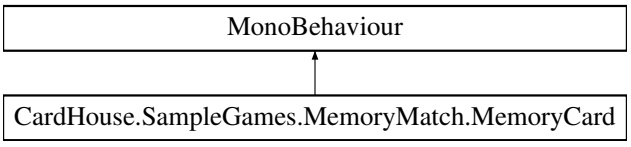
Definition at line 12 of file MatureCropDragGate.cs.

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

- MatureCropDragGate.cs

6.81 CardHouse.SampleGames.MemoryMatch.MemoryCard Class Reference

Inheritance diagram for CardHouse.SampleGames.MemoryMatch.MemoryCard:



Public Member Functions

- void Apply (Sprite sprite)
- void OnFlippedUp ()

Public Attributes

- SpriteRenderer MySpriteRenderer
- Sprite MySprite

6.81.1 Detailed Description

Definition at line 5 of file MemoryCard.cs.

6.81.2 Member Function Documentation

6.81.2.1 Apply()

```
void CardHouse.SampleGames.MemoryMatch.MemoryCard.Apply (
            Sprite sprite )
```

Definition at line 23 of file MemoryCard.cs.

6.81.2.2 OnFlippedUp()

```
void CardHouse.SampleGames.MemoryMatch.MemoryCard.OnFlippedUp ( )
```

Definition at line 29 of file MemoryCard.cs.

6.81.3 Member Data Documentation

6.81.3.1 MySprite

```
Sprite CardHouse.SampleGames.MemoryMatch.MemoryCard.MySprite
```

Definition at line 9 of file MemoryCard.cs.

6.81.3.2 MySpriteRenderer

```
SpriteRenderer CardHouse.SampleGames.MemoryMatch.MemoryCard.MySpriteRenderer
```

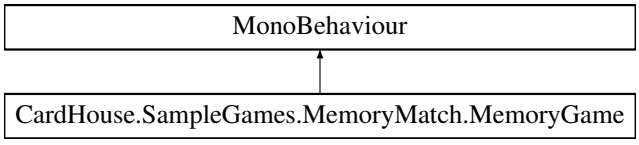
Definition at line 7 of file MemoryCard.cs.

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

- MemoryCard.cs

6.82 CardHouse.SampleGames.MemoryMatch.MemoryGame Class Reference

Inheritance diagram for CardHouse.SampleGames.MemoryMatch.MemoryGame:



Public Member Functions

- void Restart ()
- void Flip (MemoryCard card)

Public Attributes

- GameObject CardPrefab
- List< CardGroup > Slots
- List< Sprite > Sprites
- MemoryUI MyUI
- GameObject MatchEffect

Static Public Attributes

- static MemoryGame Instance

6.82.1 Detailed Description

Definition at line 8 of file MemoryGame.cs.

6.82.2 Member Function Documentation

6.82.2.1 Flip()

```
void CardHouse.SampleGames.MemoryMatch.MemoryGame.Flip (
            MemoryCard card )
```

Definition at line 91 of file MemoryGame.cs.

6.82.2.2 Restart()

```
void CardHouse.SampleGames.MemoryMatch.MemoryGame.Restart ( )
```

Definition at line 30 of file MemoryGame.cs.

6.82.3 Member Data Documentation

6.82.3.1 CardPrefab

```
GameObject CardHouse.SampleGames.MemoryMatch.MemoryGame.CardPrefab
```

Definition at line 10 of file MemoryGame.cs.

6.82.3.2 Instance

```
MemoryGame CardHouse.SampleGames.MemoryMatch.MemoryGame.Instance  [static]
```

Definition at line 23 of file MemoryGame.cs.

6.82.3.3 MatchEffect

```
GameObject CardHouse.SampleGames.MemoryMatch.MemoryGame.MatchEffect
```

Definition at line 15 of file MemoryGame.cs.

6.82.3.4 MyUI

```
MemoryUI CardHouse.SampleGames.MemoryMatch.MemoryGame.MyUI
```

Definition at line 14 of file MemoryGame.cs.

6.82.3.5 Slots

```
List<CardGroup> CardHouse.SampleGames.MemoryMatch.MemoryGame.Slots
```

Definition at line 11 of file MemoryGame.cs.

6.82.3.6 Sprites

```
List<Sprite> CardHouse.SampleGames.MemoryMatch.MemoryGame.Sprites
```

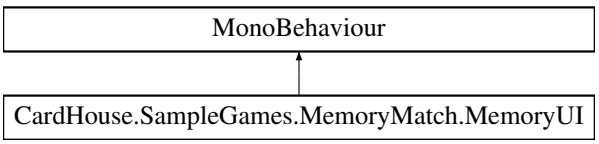
Definition at line 12 of file MemoryGame.cs.

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

- MemoryGame.cs

6.83 CardHouse.SampleGames.MemoryMatch.MemoryUI Class Reference

Inheritance diagram for CardHouse.SampleGames.MemoryMatch.MemoryUI:



Public Member Functions

- void UpdateMatches (int matches)
- void UpdateTimer (float timer)

Public Attributes

- Text TimerText
- Text MatchText

6.83.1 Detailed Description

Definition at line 6 of file MemoryUI.cs.

6.83.2 Member Function Documentation

6.83.2.1 UpdateMatches()

```
void CardHouse.SampleGames.MemoryMatch.MemoryUI.UpdateMatches (
            int matches )
```

Definition at line 11 of file MemoryUI.cs.

6.83.2.2 UpdateTimer()

```
void CardHouse.SampleGames.MemoryMatch.MemoryUI.UpdateTimer (
            float timer )
```

Definition at line 16 of file MemoryUI.cs.

6.83.3 Member Data Documentation

6.83.3.1 MatchText

```
Text CardHouse.SampleGames.MemoryMatch.MemoryUI.MatchText
```

Definition at line 9 of file MemoryUI.cs.

6.83.3.2 TimerText

```
Text CardHouse.SampleGames.MemoryMatch.MemoryUI.TimerText
```

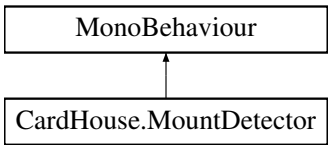
Definition at line 8 of file MemoryUI.cs.

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

- MemoryUI.cs

6.84 CardHouse.MountDetector Class Reference

Inheritance diagram for CardHouse.MountDetector:



Public Attributes

- UnityEvent OnMount

6.84.1 Detailed Description

Definition at line 7 of file MountDetector.cs.

6.84.2 Member Data Documentation

6.84.2.1 OnMount

```
UnityEvent CardHouse.MountDetector.OnMount
```

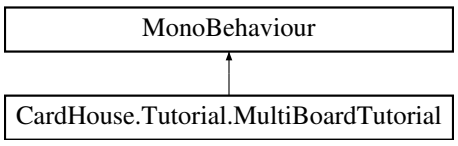
Definition at line 9 of file MountDetector.cs.

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

- MountDetector.cs

6.85 CardHouse.Tutorial.MultiBoardTutorial Class Reference

Inheritance diagram for CardHouse.Tutorial.MultiBoardTutorial:



Classes

- class InstructionImagePair

Public Member Functions

- void InstructionsForward ()
- void InstructionsBackward ()
- void SetupBoard ()
- void UpdatePlayerCount ()
- void UpdateSpacingMultiplier ()

Public Attributes

- MultiplayerBoardSetup SetupScript
- TMP_Text PlayerCountLabel
- Slider PlayerCountSlider
- TMP_Text SpacingLabel
- Slider SpacingSlider
- Button SetupButton
- GameObject InstructionsRoot
- Image InstructionsImage
- TMP_Text InstructionsText
- List< InstructionImagePair > Instructions
- TMP_Text PageNumberText
- Button ForwardButton
- Button BackButton
- int InstructionIndex
- GameObject CommonArea

6.85.1 Detailed Description

Definition at line 9 of file MultiBoardTutorial.cs.

6.85.2 Member Function Documentation

6.85.2.1 InstructionsBackward()

```
void CardHouse.Tutorial.MultiBoardTutorial.InstructionsBackward ( )
```

Definition at line 122 of file MultiBoardTutorial.cs.

6.85.2.2 InstructionsForward()

```
void CardHouse.Tutorial.MultiBoardTutorial.InstructionsForward ( )
```

Definition at line 108 of file MultiBoardTutorial.cs.

6.85.2.3 SetupBoard()

```
void CardHouse.Tutorial.MultiBoardTutorial.SetupBoard ( )
```

Definition at line 129 of file MultiBoardTutorial.cs.

6.85.2.4 UpdatePlayerCount()

```
void CardHouse.Tutorial.MultiBoardTutorial.UpdatePlayerCount ( )
```

Definition at line 137 of file MultiBoardTutorial.cs.

6.85.2.5 UpdateSpacingMultiplier()

```
void CardHouse.Tutorial.MultiBoardTutorial.UpdateSpacingMultiplier ( )
```

Definition at line 142 of file MultiBoardTutorial.cs.

6.85.3 Member Data Documentation

6.85.3.1 BackButton

```
Button CardHouse.Tutorial.MultiBoardTutorial.BackButton
```

Definition at line 32 of file MultiBoardTutorial.cs.

6.85.3.2 CommonArea

```
GameObject CardHouse.Tutorial.MultiBoardTutorial.CommonArea
```

Definition at line 35 of file MultiBoardTutorial.cs.

6.85.3.3 ForwardButton

```
Button CardHouse.Tutorial.MultiBoardTutorial.ForwardButton
```

Definition at line 31 of file MultiBoardTutorial.cs.

6.85.3.4 InstructionIndex

```
int CardHouse.Tutorial.MultiBoardTutorial.InstructionIndex
```

Definition at line 33 of file MultiBoardTutorial.cs.

6.85.3.5 Instructions

```
List<InstructionImagePair> CardHouse.Tutorial.MultiBoardTutorial.Instructions
```

Definition at line 29 of file MultiBoardTutorial.cs.

6.85.3.6 InstructionsImage

```
Image CardHouse.Tutorial.MultiBoardTutorial.InstructionsImage
```

Definition at line 27 of file MultiBoardTutorial.cs.

6.85.3.7 InstructionsRoot

```
GameObject CardHouse.Tutorial.MultiBoardTutorial.InstructionsRoot
```

Definition at line 26 of file MultiBoardTutorial.cs.

6.85.3.8 InstructionsText

```
TMP_Text CardHouse.Tutorial.MultiBoardTutorial.InstructionsText
```

Definition at line 28 of file MultiBoardTutorial.cs.

6.85.3.9 PageNumberText

```
TMP_Text CardHouse.Tutorial.MultiBoardTutorial.PageNumberText
```

Definition at line 30 of file MultiBoardTutorial.cs.

6.85.3.10 PlayerCountLabel

```
TMP_Text CardHouse.Tutorial.MultiBoardTutorial.PlayerCountLabel
```

Definition at line 20 of file MultiBoardTutorial.cs.

6.85.3.11 PlayerCountSlider

```
Slider CardHouse.Tutorial.MultiBoardTutorial.PlayerCountSlider
```

Definition at line 21 of file MultiBoardTutorial.cs.

6.85.3.12 SetupButton

```
Button CardHouse.Tutorial.MultiBoardTutorial.SetupButton
```

Definition at line 24 of file MultiBoardTutorial.cs.

6.85.3.13 SetupScript

```
MultiplayerBoardSetup CardHouse.Tutorial.MultiBoardTutorial.SetupScript
```

Definition at line 19 of file MultiBoardTutorial.cs.

6.85.3.14 SpacingLabel

```
TMP_Text CardHouse.Tutorial.MultiBoardTutorial.SpacingLabel
```

Definition at line 22 of file MultiBoardTutorial.cs.

6.85.3.15 SpacingSlider

```
Slider CardHouse.Tutorial.MultiBoardTutorial.SpacingSlider
```

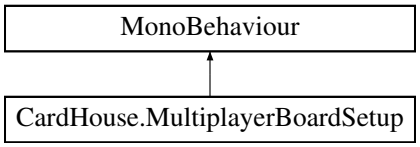
Definition at line 23 of file MultiBoardTutorial.cs.

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

- MultiBoardTutorial.cs

6.86 CardHouse.MultiplayerBoardSetup Class Reference

Inheritance diagram for CardHouse.MultiplayerBoardSetup:



Classes

- class GroupTransitionByName

Public Types

- enum **PvpMode** { **PlayerToEnemy** , **EnemyToPlayer** }

Public Member Functions

- GameObject[ ] GetAllBoards ()
- void Setup (bool callSetupScripts=true)

Public Attributes

- bool RunOnStart = true
- GameObject PlayerBoard
- int PlayerCount = 2
- float SpacingMultiplier = 1.0f
- List< GroupTransitionByName > PlayerToPlayerInteractions

6.86.1 Detailed Description

Definition at line 10 of file MultiplayerBoardSetup.cs.

6.86.2 Member Enumeration Documentation

6.86.2.1 PvpMode

```
enum CardHouse.MultiplayerBoardSetup.PvpMode
```

Definition at line 22 of file MultiplayerBoardSetup.cs.

6.86.3 Member Function Documentation

6.86.3.1 GetAllBoards()

```
GameObject[] CardHouse.MultiplayerBoardSetup.GetAllBoards ( )
```

Definition at line 44 of file MultiplayerBoardSetup.cs.

6.86.3.2 Setup()

```
void CardHouse.MultiplayerBoardSetup.Setup (
            bool callSetupScripts = true )
```

Definition at line 59 of file MultiplayerBoardSetup.cs.

6.86.4 Member Data Documentation

6.86.4.1 PlayerBoard

```
GameObject CardHouse.MultiplayerBoardSetup.PlayerBoard
```

Definition at line 29 of file MultiplayerBoardSetup.cs.

6.86.4.2 PlayerCount

```
int CardHouse.MultiplayerBoardSetup.PlayerCount = 2
```

Definition at line 30 of file MultiplayerBoardSetup.cs.

6.86.4.3 PlayerToPlayerInteractions

```
List<GroupTransitionByName> CardHouse.MultiplayerBoardSetup.PlayerToPlayerInteractions
```

Definition at line 33 of file MultiplayerBoardSetup.cs.

6.86.4.4 RunOnStart

```
bool CardHouse.MultiplayerBoardSetup.RunOnStart = true
```

Definition at line 28 of file MultiplayerBoardSetup.cs.

6.86.4.5 SpacingMultiplier

```
float CardHouse.MultiplayerBoardSetup.SpacingMultiplier = 1.0f
```

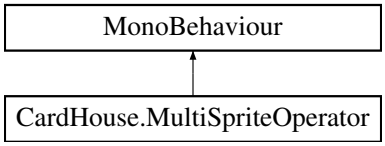
Definition at line 31 of file MultiplayerBoardSetup.cs.

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

- MultiplayerBoardSetup.cs

6.87 CardHouse.MultiSpriteOperator Class Reference

Inheritance diagram for CardHouse.MultiSpriteOperator:



Public Member Functions

- void Activate (string name)
- void Activate (string name, Object voter)
- void Remove (Object voter)
- void RemoveVote ()

Public Attributes

- List< SpriteOperator > SpriteOperators

6.87.1 Detailed Description

Definition at line 6 of file MultiSpriteOperator.cs.

6.87.2 Member Function Documentation

6.87.2.1 Activate() [1/2]

```
void CardHouse.MultiSpriteOperator.Activate (
            string name )
```

Definition at line 10 of file MultiSpriteOperator.cs.

6.87.2.2 Activate() [2/2]

```
void CardHouse.MultiSpriteOperator.Activate (
            string name,
            Object voter )
```

Definition at line 15 of file MultiSpriteOperator.cs.

6.87.2.3 Remove()

```
void CardHouse.MultiSpriteOperator.Remove (
            Object voter )
```

Definition at line 23 of file MultiSpriteOperator.cs.

6.87.2.4 RemoveVote()

```
void CardHouse.MultiSpriteOperator.RemoveVote ( )
```

Definition at line 31 of file MultiSpriteOperator.cs.

6.87.3 Member Data Documentation

6.87.3.1 SpriteOperators

```
List<SpriteOperator> CardHouse.MultiSpriteOperator.SpriteOperators
```

Definition at line 8 of file MultiSpriteOperator.cs.

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

- MultiSpriteOperator.cs

6.88 CardHouse.SpriteColorOperator.NamedColor Class Reference

Public Attributes

- string Name
- Color Color

6.88.1 Detailed Description

Definition at line 10 of file SpriteColorOperator.cs.

6.88.2 Member Data Documentation

6.88.2.1 Color

```
Color CardHouse.SpriteColorOperator.NamedColor.Color
```

Definition at line 13 of file SpriteColorOperator.cs.

6.88.2.2 Name

```
string CardHouse.SpriteColorOperator.NamedColor.Name
```

Definition at line 12 of file SpriteColorOperator.cs.

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

- SpriteColorOperator.cs

6.89 CardHouse.GroupRegistry.NamedGroup Class Reference

Public Attributes

- int PlayerIndex
- GroupName Name
- CardGroup Group

6.89.1 Detailed Description

Definition at line 10 of file GroupRegistry.cs.

6.89.2 Member Data Documentation

6.89.2.1 Group

```
CardGroup CardHouse.GroupRegistry.NamedGroup.Group
```

Definition at line 14 of file GroupRegistry.cs.

6.89.2.2 Name

```
GroupName CardHouse.GroupRegistry.NamedGroup.Name
```

Definition at line 13 of file GroupRegistry.cs.

6.89.2.3 PlayerIndex

```
int CardHouse.GroupRegistry.NamedGroup.PlayerIndex
```

Definition at line 12 of file GroupRegistry.cs.

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

- GroupRegistry.cs

6.90 CardHouse.SpriteImageOperator.NamedSprite Class Reference

Public Attributes

- string Name
- Sprite Sprite

6.90.1 Detailed Description

Definition at line 10 of file SpriteImageOperator.cs.

6.90.2 Member Data Documentation

6.90.2.1 Name

```
string CardHouse.SpriteImageOperator.NamedSprite.Name
```

Definition at line 12 of file SpriteImageOperator.cs.

6.90.2.2 Sprite

```
Sprite CardHouse.SpriteImageOperator.NamedSprite.Sprite
```

Definition at line 13 of file SpriteImageOperator.cs.

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

- SpriteImageOperator.cs

6.91 CardHouse.NoParams Class Reference

6.91.1 Detailed Description

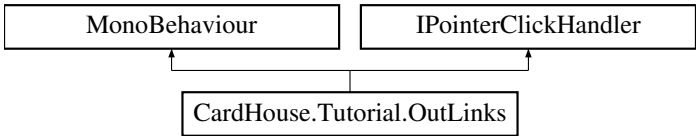
Definition at line 3 of file NoParams.cs.

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

- NoParams.cs

6.92 CardHouse.Tutorial.OutLinks Class Reference

Inheritance diagram for CardHouse.Tutorial.OutLinks:



Public Member Functions

- void OnPointerClick (PointerEventData eventData)

Public Attributes

- TMP_Text Text

6.92.1 Detailed Description

Definition at line 7 of file OutLinks.cs.

6.92.2 Member Function Documentation

6.92.2.1 OnPointerClick()

```
void CardHouse.Tutorial.OutLinks.OnPointerClick (
            PointerEventData eventData )
```

Definition at line 11 of file OutLinks.cs.

6.92.3 Member Data Documentation

6.92.3.1 Text

```
TMP_Text CardHouse.Tutorial.OutLinks.Text
```

Definition at line 9 of file OutLinks.cs.

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

- OutLinks.cs

6.93 CardHouse.Phase Class Reference

Public Member Functions

- IEnumerator Start ()
- IEnumerator End ()
- bool IsValidDragStart (CardGroup source, DragAction dragAction)
- bool IsValidDrag (CardGroup source, CardGroup destination, DragAction dragAction)

Public Attributes

- string Name
- int PlayerIndex
- Transform CameraPosition
- Transform CardPresentationPosition
- List< Button > ActiveButtons
- List< ClickDetector > ValidClickTargets
- List< DragTransition > ValidDrags
- List< TimedEvent > OnPhaseStartEventChain
- List< TimedEvent > OnPhaseEndEventChain

6.93.1 Detailed Description

Definition at line 11 of file Phase.cs.

6.93.2 Member Function Documentation

6.93.2.1 End()

```
IEnumerator CardHouse.Phase.End ( )
```

Definition at line 33 of file Phase.cs.

6.93.2.2 IsValidDrag()

```
bool CardHouse.Phase.IsValidDrag (
            CardGroup source,
            CardGroup destination,
            DragAction dragAction )
```

Definition at line 43 of file Phase.cs.

6.93.2.3 IsValidDragStart()

```
bool CardHouse.Phase.IsValidDragStart (
            CardGroup source,
            DragAction dragAction )
```

Definition at line 38 of file Phase.cs.

6.93.2.4 Start()

```
IEnumerator CardHouse.Phase.Start ( )
```

Definition at line 23 of file Phase.cs.

6.93.3 Member Data Documentation

6.93.3.1 ActiveButtons

```
List<Button> CardHouse.Phase.ActiveButtons
```

Definition at line 17 of file Phase.cs.

6.93.3.2 CameraPosition

```
Transform CardHouse.Phase.CameraPosition
```

Definition at line 15 of file Phase.cs.

6.93.3.3 CardPresentationPosition

```
Transform CardHouse.Phase.CardPresentationPosition
```

Definition at line 16 of file Phase.cs.

6.93.3.4 Name

```
string CardHouse.Phase.Name
```

Definition at line 13 of file Phase.cs.

6.93.3.5 OnPhaseEndEventChain

```
List<TimedEvent> CardHouse.Phase.OnPhaseEndEventChain
```

Definition at line 21 of file Phase.cs.

6.93.3.6 OnPhaseStartEventChain

```
List<TimedEvent> CardHouse.Phase.OnPhaseStartEventChain
```

Definition at line 20 of file Phase.cs.

6.93.3.7 PlayerIndex

```
int CardHouse.Phase.PlayerIndex
```

Definition at line 14 of file Phase.cs.

6.93.3.8 ValidClickTargets

```
List<ClickDetector> CardHouse.Phase.ValidClickTargets
```

Definition at line 18 of file Phase.cs.

6.93.3.9 ValidDrags

```
List<DragTransition> CardHouse.Phase.ValidDrags
```

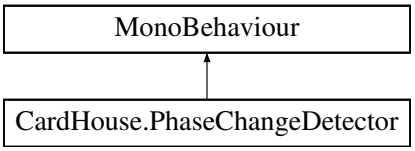
Definition at line 19 of file Phase.cs.

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

- Phase.cs

6.94 CardHouse.PhaseChangeDetector Class Reference

Inheritance diagram for CardHouse.PhaseChangeDetector:



Public Attributes

- UnityEvent OnPhaseChange

6.94.1 Detailed Description

Definition at line 6 of file PhaseChangeDetector.cs.

6.94.2 Member Data Documentation

6.94.2.1 OnPhaseChange

```
UnityEvent CardHouse.PhaseChangeDetector.OnPhaseChange
```

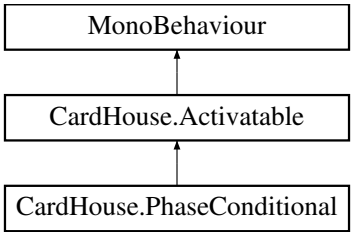
Definition at line 8 of file PhaseChangeDetector.cs.

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

- PhaseChangeDetector.cs

6.95 CardHouse.PhaseConditional Class Reference

Inheritance diagram for CardHouse.PhaseConditional:



Public Attributes

- List< StringUnityActionKvp > Responses

Protected Member Functions

- override void OnActivate ()
- virtual void OnActivate ()

Additional Inherited Members

Public Member Functions inherited from CardHouse.Activatable

- void Activate ()

6.95.1 Detailed Description

Definition at line 7 of file PhaseConditional.cs.

6.95.2 Member Function Documentation

6.95.2.1 OnActivate()

```
override void CardHouse.PhaseConditional.OnActivate ( )  [protected], [virtual]
```

Reimplemented from CardHouse.Activatable.

Definition at line 11 of file PhaseConditional.cs.

6.95.3 Member Data Documentation

6.95.3.1 Responses

```
List<StringUnityActionKvp> CardHouse.PhaseConditional.Responses
```

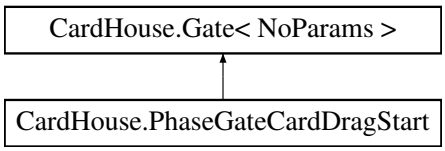
Definition at line 9 of file PhaseConditional.cs.

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

- PhaseConditional.cs

6.96 CardHouse.PhaseGateCardDragStart Class Reference

Inheritance diagram for CardHouse.PhaseGateCardDragStart:



Protected Member Functions

- override bool IsUnlockedInternal (NoParams gateParams)

Protected Member Functions inherited from CardHouse.Gate< NoParams >

- abstract bool **IsUnlockedInternal** (T argObject)

Additional Inherited Members

Public Member Functions inherited from CardHouse.Gate< NoParams >

- bool IsUnlocked (T argObject)

6.96.1 Detailed Description

Definition at line 6 of file PhaseGateCardDragStart.cs.

6.96.2 Member Function Documentation

6.96.2.1 IsUnlockedInternal()

```
override bool CardHouse.PhaseGateCardDragStart.IsUnlockedInternal (
            NoParams gateParams )  [protected]
```

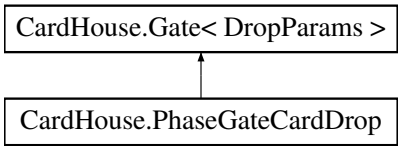
Definition at line 17 of file PhaseGateCardDragStart.cs.

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

- PhaseGateCardDragStart.cs

6.97 CardHouse.PhaseGateCardDrop Class Reference

Inheritance diagram for CardHouse.PhaseGateCardDrop:



Protected Member Functions

- override bool IsUnlockedInternal (DropParams gateParams)

Protected Member Functions inherited from CardHouse.Gate< DropParams >

- abstract bool **IsUnlockedInternal** (T argObject)

Additional Inherited Members

Public Member Functions inherited from CardHouse.Gate< DropParams >

- bool IsUnlocked (T argObject)

6.97.1 Detailed Description

Definition at line 6 of file PhaseGateCardDrop.cs.

6.97.2 Member Function Documentation

6.97.2.1 IsUnlockedInternal()

```
override bool CardHouse.PhaseGateCardDrop.IsUnlockedInternal (
            DropParams gateParams )  [protected]
```

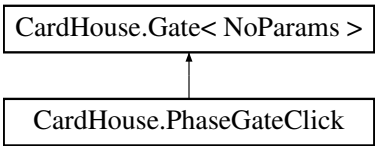
Definition at line 15 of file PhaseGateCardDrop.cs.

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

- PhaseGateCardDrop.cs

6.98 CardHouse.PhaseGateClick Class Reference

Inheritance diagram for CardHouse.PhaseGateClick:



Protected Member Functions

- override bool IsUnlockedInternal (NoParams gateParams)

Protected Member Functions inherited from CardHouse.Gate< NoParams >

- abstract bool **IsUnlockedInternal** (T argObject)

Additional Inherited Members

Public Member Functions inherited from CardHouse.Gate< NoParams >

- bool IsUnlocked (T argObject)

6.98.1 Detailed Description

Definition at line 6 of file PhaseGateClick.cs.

6.98.2 Member Function Documentation

6.98.2.1 IsUnlockedInternal()

```
override bool CardHouse.PhaseGateClick.IsUnlockedInternal (
            NoParams gateParams )  [protected]
```

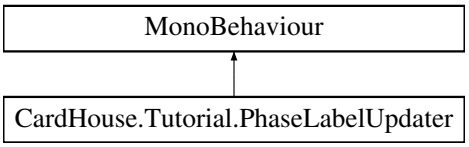
Definition at line 13 of file PhaseGateClick.cs.

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

- PhaseGateClick.cs

6.99 CardHouse.Tutorial.PhaseLabelUpdater Class Reference

Inheritance diagram for CardHouse.Tutorial.PhaseLabelUpdater:



Public Member Functions

- void UpdatePhaseLabel ()

Public Attributes

- TMP_Text PhaseText

6.99.1 Detailed Description

Definition at line 6 of file PhaseLabelUpdater.cs.

6.99.2 Member Function Documentation

6.99.2.1 UpdatePhaseLabel()

```
void CardHouse.Tutorial.PhaseLabelUpdater.UpdatePhaseLabel ( )
```

Definition at line 10 of file PhaseLabelUpdater.cs.

6.99.3 Member Data Documentation

6.99.3.1 PhaseText

```
TMP_Text CardHouse.Tutorial.PhaseLabelUpdater.PhaseText
```

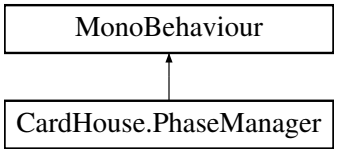
Definition at line 8 of file PhaseLabelUpdater.cs.

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

- PhaseLabelUpdater.cs

6.100 CardHouse.PhaseManager Class Reference

Inheritance diagram for CardHouse.PhaseManager:



Public Member Functions

- void HardReset ()
- void NextPhase ()
- bool IsValidDragStart (CardGroup source, DragAction dragAction)
- bool IsValidDrag (CardGroup source, CardGroup destination, DragAction dragAction)
- bool IsValidClick (ClickDetector blutton)
- void SetCameraPosition (Transform cameraPosition)

Public Attributes

- List< Button > AllPhaseDependentButtons
- List< Phase > Phases
- Action< Phase > OnPhaseChanged

Static Public Attributes

- static PhaseManager Instance

Properties

- Phase CurrentPhase [get]
- int PlayerIndex [get]

6.100.1 Detailed Description

Definition at line 9 of file PhaseManager.cs.

6.100.2 Member Function Documentation

6.100.2.1 HardReset()

```
void CardHouse.PhaseManager.HardReset ( )
```

Definition at line 37 of file PhaseManager.cs.

6.100.2.2 IsValidClick()

```
bool CardHouse.PhaseManager.IsValidClick (
            ClickDetector blutton )
```

Definition at line 86 of file PhaseManager.cs.

6.100.2.3 IsValidDrag()

```
bool CardHouse.PhaseManager.IsValidDrag (
            CardGroup source,
            CardGroup destination,
            DragAction dragAction )
```

Definition at line 78 of file PhaseManager.cs.

6.100.2.4 IsValidDragStart()

```
bool CardHouse.PhaseManager.IsValidDragStart (
            CardGroup source,
            DragAction dragAction )
```

Definition at line 70 of file PhaseManager.cs.

6.100.2.5 NextPhase()

```
void CardHouse.PhaseManager.NextPhase ( )
```

Definition at line 53 of file PhaseManager.cs.

6.100.2.6 SetCameraPosition()

```
void CardHouse.PhaseManager.SetCameraPosition (
            Transform cameraPosition )
```

Definition at line 94 of file PhaseManager.cs.

6.100.3 Member Data Documentation

6.100.3.1 AllPhaseDependentButtons

```
List<Button> CardHouse.PhaseManager.AllPhaseDependentButtons
```

Definition at line 11 of file PhaseManager.cs.

6.100.3.2 Instance

```
PhaseManager CardHouse.PhaseManager.Instance  [static]
```

Definition at line 21 of file PhaseManager.cs.

6.100.3.3 OnPhaseChanged

```
Action<Phase> CardHouse.PhaseManager.OnPhaseChanged
```

Definition at line 20 of file PhaseManager.cs.

6.100.3.4 Phases

```
List<Phase> CardHouse.PhaseManager.Phases
```

Definition at line 12 of file PhaseManager.cs.

6.100.4 Property Documentation

6.100.4.1 CurrentPhase

```
Phase CardHouse.PhaseManager.CurrentPhase  [get]
```

Definition at line 13 of file PhaseManager.cs.

6.100.4.2 PlayerIndex

```
int CardHouse.PhaseManager.PlayerIndex  [get]
```

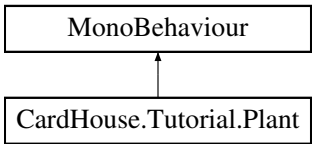
Definition at line 15 of file PhaseManager.cs.

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

- PhaseManager.cs

6.101 CardHouse.Tutorial.Plant Class Reference

Inheritance diagram for CardHouse.Tutorial.Plant:



Public Member Functions

- void Water ()
- void HideCost ()
- void Payoff ()
- bool CanBeWatered ()

Public Attributes

- TMP_Text NameText
- TMP_Text DescriptionText
- SpriteRenderer Sprite
- GameObject CostJewel
- TMP_Text CostText
- List< PlantGrowthScriptable > PossiblePlants
- int Value = 10

6.101.1 Detailed Description

Definition at line 7 of file Plant.cs.

6.101.2 Member Function Documentation

6.101.2.1 CanBeWatered()

```
bool CardHouse.Tutorial.Plant.CanBeWatered ( )
```

Definition at line 55 of file Plant.cs.

6.101.2.2 HideCost()

```
void CardHouse.Tutorial.Plant.HideCost ( )
```

Definition at line 45 of file Plant.cs.

6.101.2.3 Payoff()

```
void CardHouse.Tutorial.Plant.Payoff ( )
```

Definition at line 50 of file Plant.cs.

6.101.2.4 Water()

```
void CardHouse.Tutorial.Plant.Water ( )
```

Definition at line 28 of file Plant.cs.

6.101.3 Member Data Documentation

6.101.3.1 CostJewel

```
GameObject CardHouse.Tutorial.Plant.CostJewel
```

Definition at line 12 of file Plant.cs.

6.101.3.2 CostText

```
TMP_Text CardHouse.Tutorial.Plant.CostText
```

Definition at line 13 of file Plant.cs.

6.101.3.3 DescriptionText

```
TMP_Text CardHouse.Tutorial.Plant.DescriptionText
```

Definition at line 10 of file Plant.cs.

6.101.3.4 NameText

```
TMP_Text CardHouse.Tutorial.Plant.NameText
```

Definition at line 9 of file Plant.cs.

6.101.3.5 PossiblePlants

```
List<PlantGrowthScriptable> CardHouse.Tutorial.Plant.PossiblePlants
```

Definition at line 15 of file Plant.cs.

6.101.3.6 Sprite

```
SpriteRenderer CardHouse.Tutorial.Plant.Sprite
```

Definition at line 11 of file Plant.cs.

6.101.3.7 Value

```
int CardHouse.Tutorial.Plant.Value = 10
```

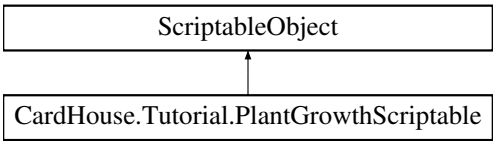
Definition at line 18 of file Plant.cs.

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

- Plant.cs

6.102 CardHouse.Tutorial.PlantGrowthScriptable Class Reference

Inheritance diagram for CardHouse.Tutorial.PlantGrowthScriptable:



Public Attributes

- List< PlantMaturityInfo > Stages

6.102.1 Detailed Description

Definition at line 7 of file PlantGrowthScriptable.cs.

6.102.2 Member Data Documentation

6.102.2.1 Stages

```
List<PlantMaturityInfo> CardHouse.Tutorial.PlantGrowthScriptable.Stages
```

Definition at line 9 of file PlantGrowthScriptable.cs.

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

- PlantGrowthScriptable.cs

6.103 CardHouse.Tutorial.PlantMaturityInfo Class Reference

Public Attributes

- string Name
- string Description
- Sprite Sprite

6.103.1 Detailed Description

Definition at line 13 of file PlantGrowthScriptable.cs.

6.103.2 Member Data Documentation

6.103.2.1 Description

```
string CardHouse.Tutorial.PlantMaturityInfo.Description
```

Definition at line 16 of file PlantGrowthScriptable.cs.

6.103.2.2 Name

```
string CardHouse.Tutorial.PlantMaturityInfo.Name
```

Definition at line 15 of file PlantGrowthScriptable.cs.

6.103.2.3 Sprite

```
Sprite CardHouse.Tutorial.PlantMaturityInfo.Sprite
```

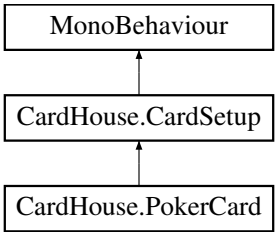
Definition at line 17 of file PlantGrowthScriptable.cs.

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

- PlantGrowthScriptable.cs

6.104 CardHouse.PokerCard Class Reference

Inheritance diagram for CardHouse.PokerCard:



Public Member Functions

- override void Apply (CardDefinition data)
- abstract void **Apply** (CardDefinition data)

Public Attributes

- SpriteRenderer Image
- SpriteRenderer BackImage

Properties

- PokerSuit Suit `[get]`
- int Rank `[get]`

6.104.1 Detailed Description

Definition at line 5 of file PokerCard.cs.

6.104.2 Member Function Documentation

6.104.2.1 Apply()

```
override void CardHouse.PokerCard.Apply (
            CardDefinition data )  [virtual]
```

Implements CardHouse.CardSetup.

Definition at line 12 of file PokerCard.cs.

6.104.3 Member Data Documentation

6.104.3.1 BackImage

```
SpriteRenderer CardHouse.PokerCard.BackImage
```

Definition at line 8 of file PokerCard.cs.

6.104.3.2 Image

```
SpriteRenderer CardHouse.PokerCard.Image
```

Definition at line 7 of file PokerCard.cs.

6.104.4 Property Documentation

6.104.4.1 Rank

```
int CardHouse.PokerCard.Rank  [get]
```

Definition at line 10 of file PokerCard.cs.

6.104.4.2 Suit

```
PokerSuit CardHouse.PokerCard.Suit  [get]
```

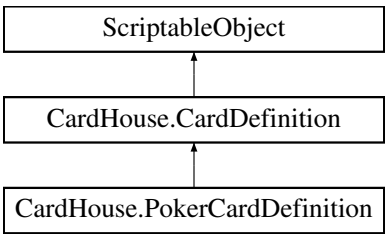
Definition at line 9 of file PokerCard.cs.

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

- PokerCard.cs

6.105 CardHouse.PokerCardDefinition Class Reference

Inheritance diagram for CardHouse.PokerCardDefinition:



Public Attributes

- int Rank
- PokerSuit Suit
- Sprite Art

Public Attributes inherited from CardHouse.CardDefinition

- Sprite BackArt

6.105.1 Detailed Description

Definition at line 6 of file PokerCardDefinition.cs.

6.105.2 Member Data Documentation

6.105.2.1 Art

```
Sprite CardHouse.PokerCardDefinition.Art
```

Definition at line 10 of file PokerCardDefinition.cs.

6.105.2.2 Rank

```
int CardHouse.PokerCardDefinition.Rank
```

Definition at line 8 of file PokerCardDefinition.cs.

6.105.2.3 Suit

```
PokerSuit CardHouse.PokerCardDefinition.Suit
```

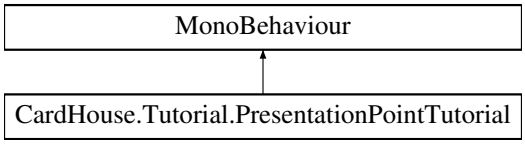
Definition at line 9 of file PokerCardDefinition.cs.

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

- PokerCardDefinition.cs

6.106 CardHouse.Tutorial.PresentationPointTutorial Class Reference

Inheritance diagram for CardHouse.Tutorial.PresentationPointTutorial:



Public Member Functions

- void Rotate (int shift)
- void Scale (int shift)
- void UpdateCameraPosition ()

Public Attributes

- int PlayerIndex
- Turning ParentTurning
- Scaling ParentScaling
- Turning ButtonParentTurning
- Scaling ButtonParentScaling
- float RotationMin
- float RotationMax
- int RotationSteps
- float ScaleMin
- float ScaleMax
- int ScaleSteps

6.106.1 Detailed Description

Definition at line 6 of file PresentationPointTutorial.cs.

6.106.2 Member Function Documentation

6.106.2.1 Rotate()

```
void CardHouse.Tutorial.PresentationPointTutorial.Rotate (
            int shift )
```

Definition at line 43 of file PresentationPointTutorial.cs.

6.106.2.2 Scale()

```
void CardHouse.Tutorial.PresentationPointTutorial.Scale (
            int shift )
```

Definition at line 50 of file PresentationPointTutorial.cs.

6.106.2.3 UpdateCameraPosition()

```
void CardHouse.Tutorial.PresentationPointTutorial.UpdateCameraPosition ( )
```

Definition at line 57 of file PresentationPointTutorial.cs.

6.106.3 Member Data Documentation

6.106.3.1 ButtonParentScaling

```
Scaling CardHouse.Tutorial.PresentationPointTutorial.ButtonParentScaling
```

Definition at line 13 of file PresentationPointTutorial.cs.

6.106.3.2 ButtonParentTurning

```
Turning CardHouse.Tutorial.PresentationPointTutorial.ButtonParentTurning
```

Definition at line 12 of file PresentationPointTutorial.cs.

6.106.3.3 ParentScaling

```
Scaling CardHouse.Tutorial.PresentationPointTutorial.ParentScaling
```

Definition at line 11 of file PresentationPointTutorial.cs.

6.106.3.4 ParentTurning

```
Turning CardHouse.Tutorial.PresentationPointTutorial.ParentTurning
```

Definition at line 10 of file PresentationPointTutorial.cs.

6.106.3.5 PlayerIndex

```
int CardHouse.Tutorial.PresentationPointTutorial.PlayerIndex
```

Definition at line 8 of file PresentationPointTutorial.cs.

6.106.3.6 RotationMax

```
float CardHouse.Tutorial.PresentationPointTutorial.RotationMax
```

Definition at line 16 of file PresentationPointTutorial.cs.

6.106.3.7 RotationMin

```
float CardHouse.Tutorial.PresentationPointTutorial.RotationMin
```

Definition at line 15 of file PresentationPointTutorial.cs.

6.106.3.8 RotationSteps

```
int CardHouse.Tutorial.PresentationPointTutorial.RotationSteps
```

Definition at line 17 of file PresentationPointTutorial.cs.

6.106.3.9 ScaleMax

```
float CardHouse.Tutorial.PresentationPointTutorial.ScaleMax
```

Definition at line 20 of file PresentationPointTutorial.cs.

6.106.3.10 ScaleMin

```
float CardHouse.Tutorial.PresentationPointTutorial.ScaleMin
```

Definition at line 19 of file PresentationPointTutorial.cs.

6.106.3.11 ScaleSteps

```
int CardHouse.Tutorial.PresentationPointTutorial.ScaleSteps
```

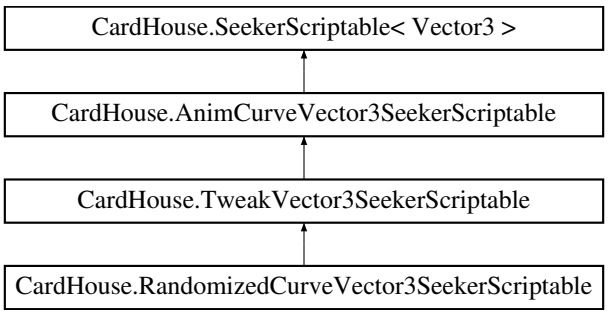
Definition at line 21 of file PresentationPointTutorial.cs.

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

- PresentationPointTutorial.cs

6.107 CardHouse.RandomizedCurveVector3SeekerScriptable Class Reference

Inheritance diagram for CardHouse.RandomizedCurveVector3SeekerScriptable:



Public Member Functions

- override Seeker< Vector3 > GetStrategy (params object[] args)

Public Member Functions inherited from CardHouse.TweakVector3SeekerScriptable

- override Seeker< Vector3 > GetStrategy (params object[] args)
- override Seeker< Vector3 > GetStrategy (params object[] args)
- abstract Seeker< T > **GetStrategy** (params object[] args)

Public Attributes

- float TweakMagnitudeMin = 1.5f
- float TweakMagnitudeMax = 2f

Public Attributes inherited from CardHouse.TweakVector3SeekerScriptable

- Vector3 Tweak
- AnimationCurve TweakMultiplier

Public Attributes inherited from CardHouse.AnimCurveVector3SeekerScriptable

- float Duration = 2f
- AnimationCurve ProgressCurve

6.107.1 Detailed Description

Definition at line 6 of file RandomizedCurveVector3SeekerScriptable.cs.

6.107.2 Member Function Documentation

6.107.2.1 GetStrategy()

```
override Seeker< Vector3 > CardHouse.RandomizedCurveVector3SeekerScriptable.GetStrategy (
            params object[] args )  [virtual]
```

Reimplemented from CardHouse.AnimCurveVector3SeekerScriptable.

Definition at line 11 of file RandomizedCurveVector3SeekerScriptable.cs.

6.107.3 Member Data Documentation

6.107.3.1 TweakMagnitudeMax

```
float CardHouse.RandomizedCurveVector3SeekerScriptable.TweakMagnitudeMax = 2f
```

Definition at line 9 of file RandomizedCurveVector3SeekerScriptable.cs.

6.107.3.2 TweakMagnitudeMin

```
float CardHouse.RandomizedCurveVector3SeekerScriptable.TweakMagnitudeMin = 1.5f
```

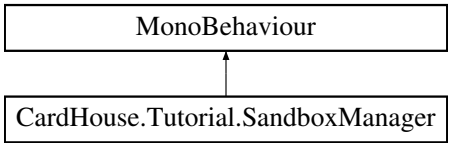
Definition at line 8 of file RandomizedCurveVector3SeekerScriptable.cs.

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

- RandomizedCurveVector3SeekerScriptable.cs

6.108 CardHouse.Tutorial.SandboxManager Class Reference

Inheritance diagram for CardHouse.Tutorial.SandboxManager:



Public Member Functions

- void Start ()
- void Reset ()
- void GoToNext ()
- void GoToPrevious ()
- void GoTo (string name)
- void ToggleSidebar ()

Public Attributes

- GameObject TutorialButtonPrefab
- Transform TutorialListRoot
- StringListScriptable Tutorials
- Animator SidebarAnimator
- TMP_Text TitleText
- GameObject NextButton
- GameObject PreviousButton
- GameObject ResetButton

Static Public Attributes

- static MultiBoardTutorial MultiBoardTutorial

6.108.1 Detailed Description

Definition at line 7 of file SandboxManager.cs.

6.108.2 Member Function Documentation

6.108.2.1 GoTo()

```
void CardHouse.Tutorial.SandboxManager.GoTo (
            string name )
```

Definition at line 76 of file SandboxManager.cs.

6.108.2.2 GoToNext()

```
void CardHouse.Tutorial.SandboxManager.GoToNext ( )
```

Definition at line 44 of file SandboxManager.cs.

6.108.2.3 GoToPrevious()

```
void CardHouse.Tutorial.SandboxManager.GoToPrevious ( )
```

Definition at line 60 of file SandboxManager.cs.

6.108.2.4 Reset()

```
void CardHouse.Tutorial.SandboxManager.Reset ( )
```

Definition at line 39 of file SandboxManager.cs.

6.108.2.5 Start()

```
void CardHouse.Tutorial.SandboxManager.Start ( )
```

Definition at line 21 of file SandboxManager.cs.

6.108.2.6 ToggleSidebar()

```
void CardHouse.Tutorial.SandboxManager.ToggleSidebar ( )
```

Definition at line 91 of file SandboxManager.cs.

6.108.3 Member Data Documentation

6.108.3.1 MultiBoardTutorial

```
MultiBoardTutorial CardHouse.Tutorial.SandboxManager.MultiBoardTutorial  [static]
```

Definition at line 19 of file SandboxManager.cs.

6.108.3.2 NextButton

```
GameObject CardHouse.Tutorial.SandboxManager.NextButton
```

Definition at line 15 of file SandboxManager.cs.

6.108.3.3 PreviousButton

```
GameObject CardHouse.Tutorial.SandboxManager.PreviousButton
```

Definition at line 16 of file SandboxManager.cs.

6.108.3.4 ResetButton

```
GameObject CardHouse.Tutorial.SandboxManager.ResetButton
```

Definition at line 17 of file SandboxManager.cs.

6.108.3.5 SidebarAnimator

```
Animator CardHouse.Tutorial.SandboxManager.SidebarAnimator
```

Definition at line 13 of file SandboxManager.cs.

6.108.3.6 TitleText

```
TMP_Text CardHouse.Tutorial.SandboxManager.TitleText
```

Definition at line 14 of file SandboxManager.cs.

6.108.3.7 TutorialButtonPrefab

```
GameObject CardHouse.Tutorial.SandboxManager.TutorialButtonPrefab
```

Definition at line 9 of file SandboxManager.cs.

6.108.3.8 TutorialListRoot

```
Transform CardHouse.Tutorial.SandboxManager.TutorialListRoot
```

Definition at line 10 of file SandboxManager.cs.

6.108.3.9 Tutorials

```
StringListScriptable CardHouse.Tutorial.SandboxManager.Tutorials
```

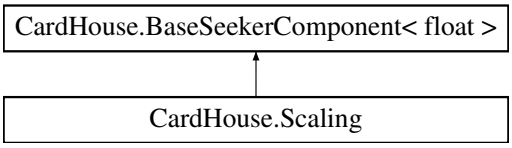
Definition at line 11 of file SandboxManager.cs.

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

- SandboxManager.cs

6.109 CardHouse.Scaling Class Reference

Inheritance diagram for CardHouse.Scaling:



Protected Member Functions

- override Seeker< float > GetDefaultSeeker ()
- override float GetCurrentValue ()
- override void SetNewValue (float value)

Protected Member Functions inherited from CardHouse.BaseSeekerComponent< float >

- abstract Seeker< T > **GetDefaultSeeker** ()
- abstract T **GetCurrentValue** ()
- abstract void **SetNewValue** (T value)

Additional Inherited Members

Public Member Functions inherited from CardHouse.BaseSeekerComponent< float >

- void StartSeeking (T destination, Seeker< T > strategy=null, bool useLocalSpace=false)

Public Attributes inherited from CardHouse.BaseSeekerComponent< float >

- SeekerScriptable< T > Strategy

Protected Attributes inherited from CardHouse.BaseSeekerComponent< float >

- Seeker< T > MyStrategy
- bool IsSeeking
- bool UseLocalSpace

6.109.1 Detailed Description

Definition at line 6 of file Scaling.cs.

6.109.2 Member Function Documentation

6.109.2.1 GetCurrentValue()

```
override float CardHouse.Scaling.GetCurrentValue ( )  [protected], [virtual]
```

Implements CardHouse.BaseSeekerComponent< float >.

Definition at line 13 of file Scaling.cs.

6.109.2.2 GetDefaultSeeker()

```
override Seeker< float > CardHouse.Scaling.GetDefaultSeeker ( )  [protected], [virtual]
```

Implements CardHouse.BaseSeekerComponent< float >.

Definition at line 8 of file Scaling.cs.

6.109.2.3 SetNewValue()

```
override void CardHouse.Scaling.SetNewValue (
            float value )  [protected]
```

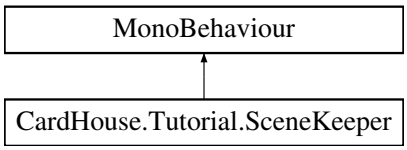
Definition at line 18 of file Scaling.cs.

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

- Scaling.cs

6.110 CardHouse.Tutorial.SceneKeeper Class Reference

Inheritance diagram for CardHouse.Tutorial.SceneKeeper:



6.110.1 Detailed Description

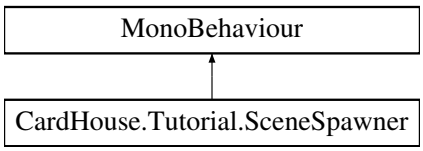
Definition at line 5 of file SceneKeeper.cs.

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

- SceneKeeper.cs

6.111 CardHouse.Tutorial.SceneSpawner Class Reference

Inheritance diagram for CardHouse.Tutorial.SceneSpawner:



Public Attributes

- string SceneToSpawn

6.111.1 Detailed Description

Definition at line 6 of file SceneSpawner.cs.

6.111.2 Member Data Documentation

6.111.2.1 SceneToSpawn

```
string CardHouse.Tutorial.SceneSpawner.SceneToSpawn
```

Definition at line 8 of file SceneSpawner.cs.

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

- SceneSpawner.cs

6.112 CardHouse.Seeker< T > Class Template Reference

Public Member Functions

- abstract Seeker< T > **MakeCopy** ()
- void StartSeeking (T from, T to)
- abstract T **Pump** (T currentValue, float TimeSinceLastFrame)
- abstract bool **IsDone** (T currentValue)

Public Attributes

- T End

Protected Attributes

- T Start

6.112.1 Detailed Description

Definition at line 3 of file Seeker.cs.

6.112.2 Member Function Documentation

6.112.2.1 StartSeeking()

```
void CardHouse.Seeker< T >.StartSeeking (
            T from,
            T to )
```

Definition at line 10 of file Seeker.cs.

6.112.3 Member Data Documentation

6.112.3.1 End

```
T CardHouse.Seeker< T >.End
```

Definition at line 6 of file Seeker.cs.

6.112.3.2 Start

```
T CardHouse.Seeker< T >.Start  [protected]
```

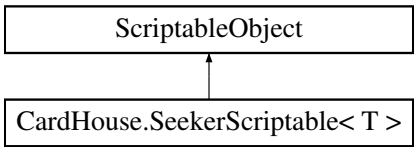
Definition at line 5 of file Seeker.cs.

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

- Seeker.cs

6.113 CardHouse.SeekerScriptable< T > Class Template Reference

Inheritance diagram for CardHouse.SeekerScriptable< T >:



Public Member Functions

- abstract Seeker< T > **GetStrategy** (params object[ ] args)

6.113.1 Detailed Description

Definition at line 5 of file SeekerScriptable.cs.

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

- SeekerScriptable.cs

6.114 CardHouse.SeekerScriptableSet Class Reference

Public Attributes

- SeekerScriptable< Vector3 > Homing
- SeekerScriptable< float > Turning
- SeekerScriptable< float > Scaling

6.114.1 Detailed Description

Definition at line 7 of file SeekerScriptableSet.cs.

6.114.2 Member Data Documentation

6.114.2.1 Homing

```
SeekerScriptable<Vector3> CardHouse.SeekerScriptableSet.Homing
```

Definition at line 9 of file SeekerScriptableSet.cs.

6.114.2.2 Scaling

```
SeekerScriptable<float> CardHouse.SeekerScriptableSet.Scaling
```

Definition at line 11 of file SeekerScriptableSet.cs.

6.114.2.3 Turning

```
SeekerScriptable<float> CardHouse.SeekerScriptableSet.Turning
```

Definition at line 10 of file SeekerScriptableSet.cs.

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

- SeekerScriptableSet.cs

6.115 CardHouse.SeekerSet Class Reference

Public Attributes

- Card Card
- Seeker< Vector3 > Homing
- Seeker< float > Turning
- Seeker< float > Scaling
- float FlipSpeed = 1f

6.115.1 Detailed Description

Definition at line 5 of file SeekerSet.cs.

6.115.2 Member Data Documentation

6.115.2.1 Card

```
Card CardHouse.SeekerSet.Card
```

Definition at line 7 of file SeekerSet.cs.

6.115.2.2 FlipSpeed

```
float CardHouse.SeekerSet.FlipSpeed = 1f
```

Definition at line 11 of file SeekerSet.cs.

6.115.2.3 Homing

```
Seeker<Vector3> CardHouse.SeekerSet.Homing
```

Definition at line 8 of file SeekerSet.cs.

6.115.2.4 Scaling

```
Seeker<float> CardHouse.SeekerSet.Scaling
```

Definition at line 10 of file SeekerSet.cs.

6.115.2.5 Turning

```
Seeker<float> CardHouse.SeekerSet.Turning
```

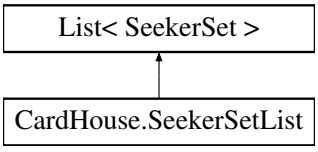
Definition at line 9 of file SeekerSet.cs.

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

- SeekerSet.cs

6.116 CardHouse.SeekerSetList Class Reference

Inheritance diagram for CardHouse.SeekerSetList:



Public Member Functions

- SeekerSet GetSeekerSetFor (Card card)

6.116.1 Detailed Description

Definition at line 6 of file SeekerSetList.cs.

6.116.2 Member Function Documentation

6.116.2.1 GetSeekerSetFor()

```
SeekerSet CardHouse.SeekerSetList.GetSeekerSetFor (
            Card card )
```

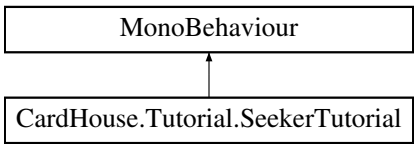
Definition at line 8 of file SeekerSetList.cs.

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

- SeekerSetList.cs

6.117 CardHouse.Tutorial.SeekerTutorial Class Reference

Inheritance diagram for CardHouse.Tutorial.SeekerTutorial:



Public Member Functions

- void Transfer (int i)

Public Attributes

- TMP_Dropdown HomingDropdown
- TMP_Dropdown TurningDropdown
- TMP_Dropdown ScalingDropdown
- List< CardGroup > Stacks
- List< StringSeekerKVP > SeekerKVPs
- Transform Waypoint

6.117.1 Detailed Description

Definition at line 8 of file SeekerTutorial.cs.

6.117.2 Member Function Documentation

6.117.2.1 Transfer()

```
void CardHouse.Tutorial.SeekerTutorial.Transfer (
            int i )
```

Definition at line 20 of file SeekerTutorial.cs.

6.117.3 Member Data Documentation

6.117.3.1 HomingDropdown

```
TMP_Dropdown CardHouse.Tutorial.SeekerTutorial.HomingDropdown
```

Definition at line 10 of file SeekerTutorial.cs.

6.117.3.2 ScalingDropdown

```
TMP_Dropdown CardHouse.Tutorial.SeekerTutorial.ScalingDropdown
```

Definition at line 12 of file SeekerTutorial.cs.

6.117.3.3 SeekerKVPs

```
List<StringSeekerKVP> CardHouse.Tutorial.SeekerTutorial.SeekerKVPs
```

Definition at line 16 of file SeekerTutorial.cs.

6.117.3.4 Stacks

```
List<CardGroup> CardHouse.Tutorial.SeekerTutorial.Stacks
```

Definition at line 14 of file SeekerTutorial.cs.

6.117.3.5 TurningDropdown

```
TMP_Dropdown CardHouse.Tutorial.SeekerTutorial.TurningDropdown
```

Definition at line 11 of file SeekerTutorial.cs.

6.117.3.6 Waypoint

```
Transform CardHouse.Tutorial.SeekerTutorial.Waypoint
```

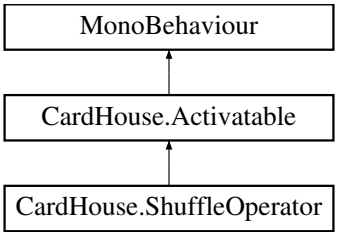
Definition at line 18 of file SeekerTutorial.cs.

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

- SeekerTutorial.cs

6.118 CardHouse.ShuffleOperator Class Reference

Inheritance diagram for CardHouse.ShuffleOperator:



Public Attributes

- List< CardGroup > GroupsToShuffleIntoDeck
- CardGroup Deck

Protected Member Functions

- override void OnActivate ()

- virtual void OnActivate ()

Additional Inherited Members

Public Member Functions inherited from CardHouse.Activatable

- void Activate ()

6.118.1 Detailed Description

Definition at line 7 of file ShuffleOperator.cs.

6.118.2 Member Function Documentation

6.118.2.1 OnActivate()

```
override void CardHouse.ShuffleOperator.OnActivate ( )  [protected], [virtual]
```

Reimplemented from CardHouse.Activatable.

Definition at line 13 of file ShuffleOperator.cs.

6.118.3 Member Data Documentation

6.118.3.1 Deck

```
CardGroup CardHouse.ShuffleOperator.Deck
```

Definition at line 11 of file ShuffleOperator.cs.

6.118.3.2 GroupsToShuffleIntoDeck

```
List<CardGroup> CardHouse.ShuffleOperator.GroupsToShuffleIntoDeck
```

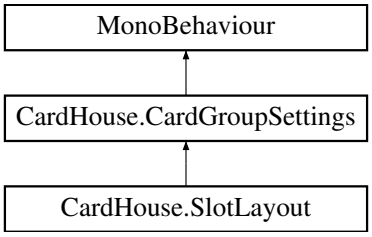
Definition at line 10 of file ShuffleOperator.cs.

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

- ShuffleOperator.cs

6.119 CardHouse.SlotLayout Class Reference

Inheritance diagram for CardHouse.SlotLayout:



Protected Member Functions

- override void ApplySpacing (List< Card > cards, SeekerSetList seekerSets=null)
- abstract void **ApplySpacing** (List< Card > cards, SeekerSetList seekerSets)

Additional Inherited Members

Public Member Functions inherited from CardHouse.CardGroupSettings

- void Apply (List< Card > cards, bool instaFlip=false, SeekerSetList seekerSets=null)

Public Attributes inherited from CardHouse.CardGroupSettings

- int CardLimit = -1
- float MountedCardAltitude = 0.01f
- CardFacing ForcedFacing
- GroupInteractability ForcedInteractability
- MountingMode DragMountingMode = MountingMode.Top
- bool UseMyScale = false

6.119.1 Detailed Description

Definition at line 6 of file SlotLayout.cs.

6.119.2 Member Function Documentation

6.119.2.1 ApplySpacing()

```
override void CardHouse.SlotLayout.ApplySpacing (
            List< Card > cards,
            SeekerSetList seekerSets = null )  [protected], [virtual]
```

Implements CardHouse.CardGroupSettings.

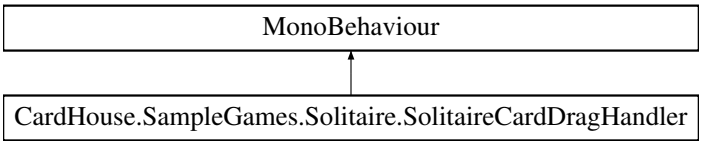
Definition at line 8 of file SlotLayout.cs.

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

- SlotLayout.cs

6.120 CardHouse.SampleGames.Solitaire.SolitaireCardDragHandler Class Reference

Inheritance diagram for CardHouse.SampleGames.Solitaire.SolitaireCardDragHandler:



Public Member Functions

- void AttachChildren ()
- void DetatchChildren ()

6.120.1 Detailed Description

Definition at line 7 of file SolitaireCardDragHandler.cs.

6.120.2 Member Function Documentation

6.120.2.1 AttachChildren()

```
void CardHouse.SampleGames.Solitaire.SolitaireCardDragHandler.AttachChildren ( )
```

Definition at line 18 of file SolitaireCardDragHandler.cs.

6.120.2.2 DetatchChildren()

```
void CardHouse.SampleGames.Solitaire.SolitaireCardDragHandler.DetatchChildren ( )
```

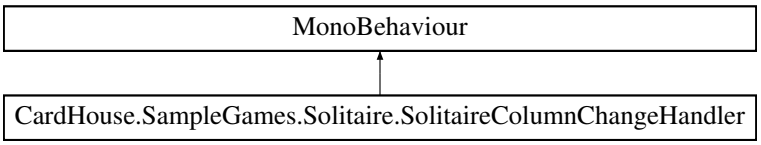
Definition at line 29 of file SolitaireCardDragHandler.cs.

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

- SolitaireCardDragHandler.cs

6.121 CardHouse.SampleGames.Solitaire.SolitaireColumnChangeHandler Class Reference

Inheritance diagram for CardHouse.SampleGames.Solitaire.SolitaireColumnChangeHandler:



Public Member Functions

- void Refresh ()

6.121.1 Detailed Description

Definition at line 6 of file SolitaireColumnChangeHandler.cs.

6.121.2 Member Function Documentation

6.121.2.1 Refresh()

```
void CardHouse.SampleGames.Solitaire.SolitaireColumnChangeHandler.Refresh ( )
```

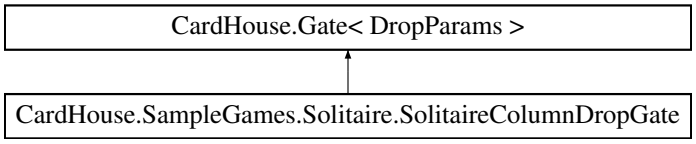
Definition at line 15 of file SolitaireColumnChangeHandler.cs.

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

- SolitaireColumnChangeHandler.cs

6.122 CardHouse.SampleGames.Solitaire.SolitaireColumnDropGate Class Reference

Inheritance diagram for CardHouse.SampleGames.Solitaire.SolitaireColumnDropGate:



Protected Member Functions

- override bool IsUnlockedInternal (DropParams gateParams)

Protected Member Functions inherited from CardHouse.Gate< DropParams >

- abstract bool **IsUnlockedInternal** (T argObject)

Additional Inherited Members

Public Member Functions inherited from CardHouse.Gate< DropParams >

- bool IsUnlocked (T argObject)

6.122.1 Detailed Description

Definition at line 5 of file SolitaireColumnDropGate.cs.

6.122.2 Member Function Documentation

6.122.2.1 IsUnlockedInternal()

```
override bool CardHouse.SampleGames.Solitaire.SolitaireColumnDropGate.IsUnlockedInternal (
            DropParams gateParams )  [protected]
```

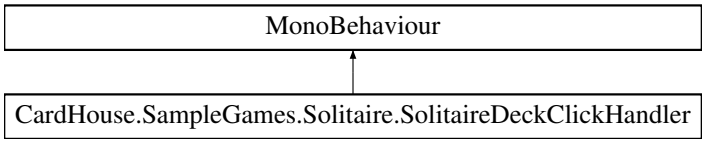
Definition at line 7 of file SolitaireColumnDropGate.cs.

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

- SolitaireColumnDropGate.cs

6.123 CardHouse.SampleGames.Solitaire.SolitaireDeckClickHandler Class Reference

Inheritance diagram for CardHouse.SampleGames.Solitaire.SolitaireDeckClickHandler:



Public Member Functions

- void FlipOrReset ()

Public Attributes

- CardTransferOperator FlipHandler
- CardTransferOperator MoveToDeckHandler
- ShuffleOperator ShuffleHandler
- CardTransferOperator DealCardHandler
- List< TimedEvent > ResetEventChain

6.123.1 Detailed Description

Definition at line 7 of file SolitaireDeckClickHandler.cs.

6.123.2 Member Function Documentation

6.123.2.1 FlipOrReset()

```
void CardHouse.SampleGames.Solitaire.SolitaireDeckClickHandler.FlipOrReset ( )
```

Definition at line 22 of file SolitaireDeckClickHandler.cs.

6.123.3 Member Data Documentation

6.123.3.1 DealCardHandler

```
CardTransferOperator CardHouse.SampleGames.Solitaire.SolitaireDeckClickHandler.DealCardHandler
```

Definition at line 12 of file SolitaireDeckClickHandler.cs.

6.123.3.2 FlipHandler

```
CardTransferOperator CardHouse.SampleGames.Solitaire.SolitaireDeckClickHandler.FlipHandler
```

Definition at line 9 of file SolitaireDeckClickHandler.cs.

6.123.3.3 MoveToDeckHandler

```
CardTransferOperator CardHouse.SampleGames.Solitaire.SolitaireDeckClickHandler.MoveToDeck↩
Handler
```

Definition at line 10 of file SolitaireDeckClickHandler.cs.

6.123.3.4 ResetEventChain

```
List<TimedEvent> CardHouse.SampleGames.Solitaire.SolitaireDeckClickHandler.ResetEventChain
```

Definition at line 13 of file SolitaireDeckClickHandler.cs.

6.123.3.5 ShuffleHandler

```
ShuffleOperator CardHouse.SampleGames.Solitaire.SolitaireDeckClickHandler.ShuffleHandler
```

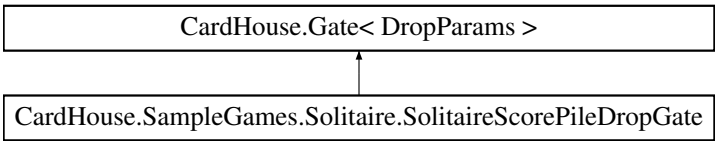
Definition at line 11 of file SolitaireDeckClickHandler.cs.

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

- SolitaireDeckClickHandler.cs

6.124 CardHouse.SampleGames.Solitaire.SolitaireScorePileDropGate Class Reference

Inheritance diagram for CardHouse.SampleGames.Solitaire.SolitaireScorePileDropGate:



Protected Member Functions

- override bool IsUnlockedInternal (DropParams gateParams)

Protected Member Functions inherited from CardHouse.Gate< DropParams >

- abstract bool **IsUnlockedInternal** (T argObject)

Additional Inherited Members

Public Member Functions inherited from CardHouse.Gate< DropParams >

- bool IsUnlocked (T argObject)

6.124.1 Detailed Description

Definition at line 3 of file SolitaireScorePileDropGate.cs.

6.124.2 Member Function Documentation

6.124.2.1 IsUnlockedInternal()

```
override bool CardHouse.SampleGames.Solitaire.SolitaireScorePileDropGate.IsUnlockedInternal (
            DropParams gateParams )  [protected]
```

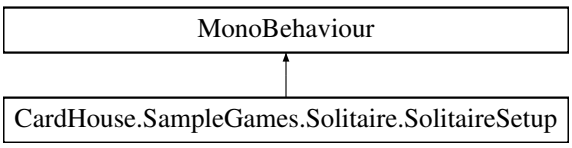
Definition at line 5 of file SolitaireScorePileDropGate.cs.

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

- SolitaireScorePileDropGate.cs

6.125 CardHouse.SampleGames.Solitaire.SolitaireSetup Class Reference

Inheritance diagram for CardHouse.SampleGames.Solitaire.SolitaireSetup:



Public Member Functions

- void TryResetBoard ()
- void DealCards ()
- void PreventReset ()
- void AllowReset ()

Public Attributes

- SeekerScriptable< Vector3 > DealingStrategy
- CardGroup Deck
- List< CardGroup > Columns
- List< CardGroup > AllGroups
- EventChain ResetBoardEventChain

6.125.1 Detailed Description

Definition at line 7 of file SolitaireSetup.cs.

6.125.2 Member Function Documentation

6.125.2.1 AllowReset()

```
void CardHouse.SampleGames.Solitaire.SolitaireSetup.AllowReset ( )
```

Definition at line 64 of file SolitaireSetup.cs.

6.125.2.2 DealCards()

```
void CardHouse.SampleGames.Solitaire.SolitaireSetup.DealCards ( )
```

Definition at line 25 of file SolitaireSetup.cs.

6.125.2.3 PreventReset()

```
void CardHouse.SampleGames.Solitaire.SolitaireSetup.PreventReset ( )
```

Definition at line 59 of file SolitaireSetup.cs.

6.125.2.4 TryResetBoard()

```
void CardHouse.SampleGames.Solitaire.SolitaireSetup.TryResetBoard ( )
```

Definition at line 17 of file SolitaireSetup.cs.

6.125.3 Member Data Documentation

6.125.3.1 AllGroups

```
List<CardGroup> CardHouse.SampleGames.Solitaire.SolitaireSetup.AllGroups
```

Definition at line 12 of file SolitaireSetup.cs.

6.125.3.2 Columns

```
List<CardGroup> CardHouse.SampleGames.Solitaire.SolitaireSetup.Columns
```

Definition at line 11 of file SolitaireSetup.cs.

6.125.3.3 DealingStrategy

```
SeekerScriptable<Vector3> CardHouse.SampleGames.Solitaire.SolitaireSetup.DealingStrategy
```

Definition at line 9 of file SolitaireSetup.cs.

6.125.3.4 Deck

```
CardGroup CardHouse.SampleGames.Solitaire.SolitaireSetup.Deck
```

Definition at line 10 of file SolitaireSetup.cs.

6.125.3.5 ResetBoardEventChain

```
EventChain CardHouse.SampleGames.Solitaire.SolitaireSetup.ResetBoardEventChain
```

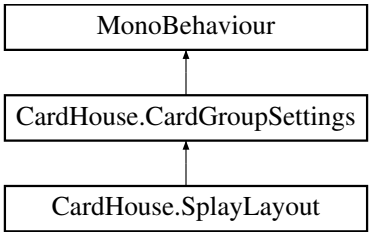
Definition at line 13 of file SolitaireSetup.cs.

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

- SolitaireSetup.cs

6.126 CardHouse.SplayLayout Class Reference

Inheritance diagram for CardHouse.SplayLayout:



Public Attributes

- float MarginalCardOffset = 0.01f
- Vector2 ArcCenterOffset = new Vector2(0f, -5f)
- float ArcMargin = 0.3f

Public Attributes inherited from CardHouse.CardGroupSettings

- int CardLimit = -1
- float MountedCardAltitude = 0.01f
- CardFacing ForcedFacing
- GroupInteractability ForcedInteractability
- MountingMode DragMountingMode = MountingMode.Top
- bool UseMyScale = false

Protected Member Functions

- override void ApplySpacing (List< Card > cards, SeekerSetList seekerSets=null)

- abstract void **ApplySpacing** (List< Card > cards, SeekerSetList seekerSets)

Additional Inherited Members

Public Member Functions inherited from CardHouse.CardGroupSettings

- void Apply (List< Card > cards, bool instaFlip=false, SeekerSetList seekerSets=null)

6.126.1 Detailed Description

Definition at line 6 of file SplayLayout.cs.

6.126.2 Member Function Documentation

6.126.2.1 ApplySpacing()

```
override void CardHouse.SplayLayout.ApplySpacing (
            List< Card > cards,
            SeekerSetList seekerSets = null )  [protected], [virtual]
```

Implements CardHouse.CardGroupSettings.

Definition at line 28 of file SplayLayout.cs.

6.126.3 Member Data Documentation

6.126.3.1 ArcCenterOffset

```
Vector2 CardHouse.SplayLayout.ArcCenterOffset = new Vector2(0f, -5f)
```

Definition at line 9 of file SplayLayout.cs.

6.126.3.2 ArcMargin

```
float CardHouse.SplayLayout.ArcMargin = 0.3f
```

Definition at line 11 of file SplayLayout.cs.

6.126.3.3 MarginalCardOffset

```
float CardHouse.SplayLayout.MarginalCardOffset = 0.01f
```

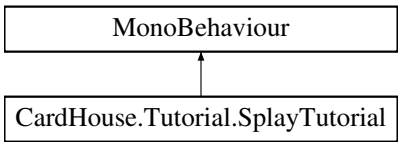
Definition at line 8 of file SplayLayout.cs.

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

- SplayLayout.cs

6.127 CardHouse.Tutorial.SplayTutorial Class Reference

Inheritance diagram for CardHouse.Tutorial.SplayTutorial:



Public Member Functions

- void AdjustXScale ()
- void AdjustArcMargin ()
- void AdjustXOffset ()
- void AdjustYOffset ()

Public Attributes

- Slider XScaleSlider
- TMP_Text XScaleText
- Slider ArcMarginSlider
- TMP_Text ArcMarginText
- Slider XOffsetSlider
- TMP_Text XOffsetText
- Slider YOffsetSlider
- TMP_Text YOffsetText
- CardGroup Deck
- SplayLayout Splay
- SpriteRenderer Reticle

6.127.1 Detailed Description

Definition at line 7 of file SplayTutorial.cs.

6.127.2 Member Function Documentation

6.127.2.1 AdjustArcMargin()

```
void CardHouse.Tutorial.SplayTutorial.AdjustArcMargin ( )
```

Definition at line 37 of file SplayTutorial.cs.

6.127.2.2 AdjustXOffset()

```
void CardHouse.Tutorial.SplayTutorial.AdjustXOffset ( )
```

Definition at line 44 of file SplayTutorial.cs.

6.127.2.3 AdjustXScale()

```
void CardHouse.Tutorial.SplayTutorial.AdjustXScale ( )
```

Definition at line 30 of file SplayTutorial.cs.

6.127.2.4 AdjustYOffset()

```
void CardHouse.Tutorial.SplayTutorial.AdjustYOffset ( )
```

Definition at line 53 of file SplayTutorial.cs.

6.127.3 Member Data Documentation

6.127.3.1 ArcMarginSlider

```
Slider CardHouse.Tutorial.SplayTutorial.ArcMarginSlider
```

Definition at line 11 of file SplayTutorial.cs.

6.127.3.2 ArcMarginText

```
TMP_Text CardHouse.Tutorial.SplayTutorial.ArcMarginText
```

Definition at line 12 of file SplayTutorial.cs.

6.127.3.3 Deck

```
CardGroup CardHouse.Tutorial.SplayTutorial.Deck
```

Definition at line 18 of file SplayTutorial.cs.

6.127.3.4 Reticle

```
SpriteRenderer CardHouse.Tutorial.SplayTutorial.Reticle
```

Definition at line 20 of file SplayTutorial.cs.

6.127.3.5 Splay

```
SplayLayout CardHouse.Tutorial.SplayTutorial.Splay
```

Definition at line 19 of file SplayTutorial.cs.

6.127.3.6 XOffsetSlider

```
Slider CardHouse.Tutorial.SplayTutorial.XOffsetSlider
```

Definition at line 13 of file SplayTutorial.cs.

6.127.3.7 XOffsetText

```
TMP_Text CardHouse.Tutorial.SplayTutorial.XOffsetText
```

Definition at line 14 of file SplayTutorial.cs.

6.127.3.8 XScaleSlider

```
Slider CardHouse.Tutorial.SplayTutorial.XScaleSlider
```

Definition at line 9 of file SplayTutorial.cs.

6.127.3.9 XScaleText

```
TMP_Text CardHouse.Tutorial.SplayTutorial.XScaleText
```

Definition at line 10 of file SplayTutorial.cs.

6.127.3.10 YOffsetSlider

```
Slider CardHouse.Tutorial.SplayTutorial.YOffsetSlider
```

Definition at line 15 of file SplayTutorial.cs.

6.127.3.11 YOffsetText

```
TMP_Text CardHouse.Tutorial.SplayTutorial.YOffsetText
```

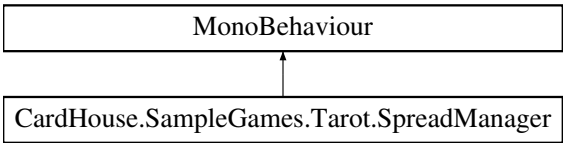
Definition at line 16 of file SplayTutorial.cs.

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

- SplayTutorial.cs

6.128 CardHouse.SampleGames.Tarot.SpreadManager Class Reference

Inheritance diagram for CardHouse.SampleGames.Tarot.SpreadManager:



Public Member Functions

- void NextSpread ()
- void PreviousSpread ()
- void ShuffleCardsBackIn ()
- void DealNextCard ()

Public Attributes

- Text SpreadLabel
- CardGroup Deck
- GameObject SpreadOrderLabelPrefab
- TMP_Text Key
- List< TarotSpread > Spreads

6.128.1 Detailed Description

Definition at line 9 of file SpreadManager.cs.

6.128.2 Member Function Documentation

6.128.2.1 DealNextCard()

```
void CardHouse.SampleGames.Tarot.SpreadManager.DealNextCard ( )
```

Definition at line 94 of file SpreadManager.cs.

6.128.2.2 NextSpread()

```
void CardHouse.SampleGames.Tarot.SpreadManager.NextSpread ( )
```

Definition at line 33 of file SpreadManager.cs.

6.128.2.3 PreviousSpread()

```
void CardHouse.SampleGames.Tarot.SpreadManager.PreviousSpread ( )
```

Definition at line 38 of file SpreadManager.cs.

6.128.2.4 ShuffleCardsBackIn()

```
void CardHouse.SampleGames.Tarot.SpreadManager.ShuffleCardsBackIn ( )
```

Definition at line 76 of file SpreadManager.cs.

6.128.3 Member Data Documentation

6.128.3.1 Deck

```
CardGroup CardHouse.SampleGames.Tarot.SpreadManager.Deck
```

Definition at line 12 of file SpreadManager.cs.

6.128.3.2 Key

```
TMP_Text CardHouse.SampleGames.Tarot.SpreadManager.Key
```

Definition at line 14 of file SpreadManager.cs.

6.128.3.3 SpreadLabel

```
Text CardHouse.SampleGames.Tarot.SpreadManager.SpreadLabel
```

Definition at line 11 of file SpreadManager.cs.

6.128.3.4 SpreadOrderLabelPrefab

```
GameObject CardHouse.SampleGames.Tarot.SpreadManager.SpreadOrderLabelPrefab
```

Definition at line 13 of file SpreadManager.cs.

6.128.3.5 Spreads

```
List<TarotSpread> CardHouse.SampleGames.Tarot.SpreadManager.Spreads
```

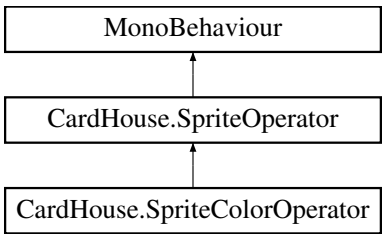
Definition at line 16 of file SpreadManager.cs.

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

- SpreadManager.cs

6.129 CardHouse.SpriteColorOperator Class Reference

Inheritance diagram for CardHouse.SpriteColorOperator:



Classes

- class NamedColor

Public Attributes

- List< NamedColor > Colors

Public Attributes inherited from CardHouse.SpriteOperator

- string FavoredState

Protected Member Functions

- override void ChangeSprite (string name)

- abstract void **ChangeSprite** (string name)

Additional Inherited Members

Public Member Functions inherited from CardHouse.SpriteOperator

- void Activate (string name)
- void Activate (string name, Object voter)
- void Remove (Object voter)

Protected Attributes inherited from CardHouse.SpriteOperator

- SpriteRenderer SpriteTarget

6.129.1 Detailed Description

Definition at line 7 of file SpriteColorOperator.cs.

6.129.2 Member Function Documentation

6.129.2.1 ChangeSprite()

```
override void CardHouse.SpriteColorOperator.ChangeSprite (
            string name )  [protected], [virtual]
```

Implements CardHouse.SpriteOperator.

Definition at line 18 of file SpriteColorOperator.cs.

6.129.3 Member Data Documentation

6.129.3.1 Colors

```
List<NamedColor> CardHouse.SpriteColorOperator.Colors
```

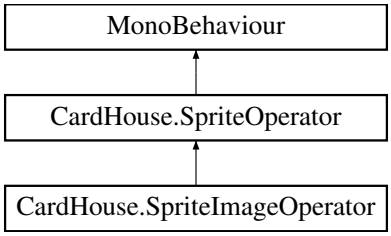
Definition at line 16 of file SpriteColorOperator.cs.

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

- SpriteColorOperator.cs

6.130 CardHouse.SpriteImageOperator Class Reference

Inheritance diagram for CardHouse.SpriteImageOperator:



Classes

- class NamedSprite

Public Attributes

- List< NamedSprite > Sprites

Public Attributes inherited from CardHouse.SpriteOperator

- string FavoredState

Protected Member Functions

- override void ChangeSprite (string name)

- abstract void **ChangeSprite** (string name)

Additional Inherited Members

Public Member Functions inherited from CardHouse.SpriteOperator

- void Activate (string name)
- void Activate (string name, Object voter)
- void Remove (Object voter)

Protected Attributes inherited from CardHouse.SpriteOperator

- SpriteRenderer SpriteTarget

6.130.1 Detailed Description

Definition at line 7 of file SpriteImageOperator.cs.

6.130.2 Member Function Documentation

6.130.2.1 ChangeSprite()

```
override void CardHouse.SpriteImageOperator.ChangeSprite (
            string name )  [protected], [virtual]
```

Implements CardHouse.SpriteOperator.

Definition at line 18 of file SpriteImageOperator.cs.

6.130.3 Member Data Documentation

6.130.3.1 Sprites

```
List<NamedSprite> CardHouse.SpriteImageOperator.Sprites
```

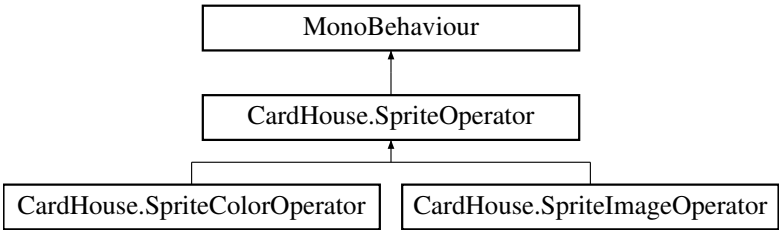
Definition at line 16 of file SpriteImageOperator.cs.

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

- SpriteImageOperator.cs

6.131 CardHouse.SpriteOperator Class Reference

Inheritance diagram for CardHouse.SpriteOperator:



Public Member Functions

- void Activate (string name)
- void Activate (string name, Object voter)
- void Remove (Object voter)

Public Attributes

- string FavoredState

Protected Member Functions

- abstract void **ChangeSprite** (string name)

Protected Attributes

- SpriteRenderer SpriteTarget

6.131.1 Detailed Description

Definition at line 8 of file SpriteOperator.cs.

6.131.2 Member Function Documentation

6.131.2.1 Activate() [1/2]

```
void CardHouse.SpriteOperator.Activate (
            string name )
```

Definition at line 19 of file SpriteOperator.cs.

6.131.2.2 Activate() [2/2]

```
void CardHouse.SpriteOperator.Activate (
            string name,
            Object voter )
```

Definition at line 24 of file SpriteOperator.cs.

6.131.2.3 Remove()

```
void CardHouse.SpriteOperator.Remove (
            Object voter )
```

Definition at line 34 of file SpriteOperator.cs.

6.131.3 Member Data Documentation

6.131.3.1 FavoredState

```
string CardHouse.SpriteOperator.FavoredState
```

Definition at line 10 of file SpriteOperator.cs.

6.131.3.2 SpriteTarget

```
SpriteRenderer CardHouse.SpriteOperator.SpriteTarget  [protected]
```

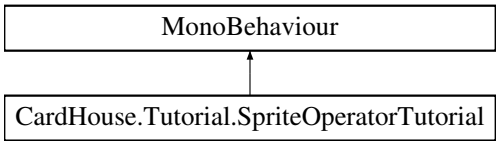
Definition at line 11 of file SpriteOperator.cs.

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

- SpriteOperator.cs

6.132 CardHouse.Tutorial.SpriteOperatorTutorial Class Reference

Inheritance diagram for CardHouse.Tutorial.SpriteOperatorTutorial:



Public Member Functions

- void RegisterColorVote (Object voter, string vote)
- void RemoveColorVote (Object voter)
- void RegisterImageVote (Object voter, string vote)
- void RemoveImageVote (Object voter)

Public Attributes

- MultiSpriteOperator ColorOperator
- SpriteImageOperator ImageOperator

Static Public Attributes

- static SpriteOperatorTutorial Instance

6.132.1 Detailed Description

Definition at line 5 of file SpriteOperatorTutorial.cs.

6.132.2 Member Function Documentation

6.132.2.1 RegisterColorVote()

```
void CardHouse.Tutorial.SpriteOperatorTutorial.RegisterColorVote (
            Object voter,
            string vote )
```

Definition at line 18 of file SpriteOperatorTutorial.cs.

6.132.2.2 RegisterImageVote()

```
void CardHouse.Tutorial.SpriteOperatorTutorial.RegisterImageVote (
            Object voter,
            string vote )
```

Definition at line 28 of file SpriteOperatorTutorial.cs.

6.132.2.3 RemoveColorVote()

```
void CardHouse.Tutorial.SpriteOperatorTutorial.RemoveColorVote (
            Object voter )
```

Definition at line 23 of file SpriteOperatorTutorial.cs.

6.132.2.4 RemoveImageVote()

```
void CardHouse.Tutorial.SpriteOperatorTutorial.RemoveImageVote (
            Object voter )
```

Definition at line 33 of file SpriteOperatorTutorial.cs.

6.132.3 Member Data Documentation

6.132.3.1 ColorOperator

```
MultiSpriteOperator CardHouse.Tutorial.SpriteOperatorTutorial.ColorOperator
```

Definition at line 7 of file SpriteOperatorTutorial.cs.

6.132.3.2 ImageOperator

```
SpriteImageOperator CardHouse.Tutorial.SpriteOperatorTutorial.ImageOperator
```

Definition at line 8 of file SpriteOperatorTutorial.cs.

6.132.3.3 Instance

```
SpriteOperatorTutorial CardHouse.Tutorial.SpriteOperatorTutorial.Instance  [static]
```

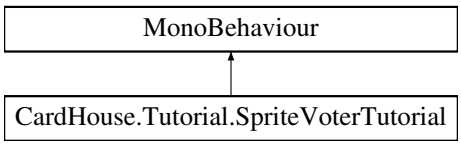
Definition at line 10 of file SpriteOperatorTutorial.cs.

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

- SpriteOperatorTutorial.cs

6.133 CardHouse.Tutorial.SpriteVoterTutorial Class Reference

Inheritance diagram for CardHouse.Tutorial.SpriteVoterTutorial:



Public Member Functions

- void OnColorDropdownUpdated ()
- void OnImageDropdownUpdated ()

Public Attributes

- UnityEvent OnStart

6.133.1 Detailed Description

Definition at line 7 of file SpriteVoterTutorial.cs.

6.133.2 Member Function Documentation

6.133.2.1 OnColorDropdownUpdated()

```
void CardHouse.Tutorial.SpriteVoterTutorial.OnColorDropdownUpdated ( )
```

Definition at line 16 of file SpriteVoterTutorial.cs.

6.133.2.2 OnImageDropdownUpdated()

```
void CardHouse.Tutorial.SpriteVoterTutorial.OnImageDropdownUpdated ( )
```

Definition at line 33 of file SpriteVoterTutorial.cs.

6.133.3 Member Data Documentation

6.133.3.1 OnStart

```
UnityEvent CardHouse.Tutorial.SpriteVoterTutorial.OnStart
```

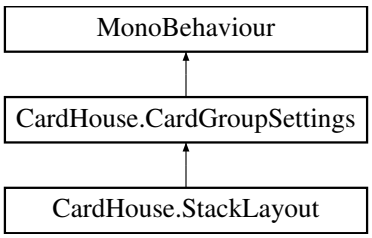
Definition at line 9 of file SpriteVoterTutorial.cs.

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

- SpriteVoterTutorial.cs

6.134 CardHouse.StackLayout Class Reference

Inheritance diagram for CardHouse.StackLayout:



Public Attributes

- Vector3 MarginalCardOffset = new Vector3(0.01f, 0.01f, -0.01f)
- TriggerEnterRelay SecondaryCollider
- bool Straighten = true

Public Attributes inherited from CardHouse.CardGroupSettings

- int CardLimit = -1
- float MountedCardAltitude = 0.01f
- CardFacing ForcedFacing
- GroupInteractability ForcedInteractability
- MountingMode DragMountingMode = MountingMode.Top
- bool UseMyScale = false

Protected Member Functions

- override void ApplySpacing (List< Card > cards, SeekerSetList seekerSets=null)

- abstract void **ApplySpacing** (List< Card > cards, SeekerSetList seekerSets)

Additional Inherited Members

Public Member Functions inherited from CardHouse.CardGroupSettings

- void Apply (List< Card > cards, bool instaFlip=false, SeekerSetList seekerSets=null)

6.134.1 Detailed Description

Definition at line 6 of file StackLayout.cs.

6.134.2 Member Function Documentation

6.134.2.1 ApplySpacing()

```
override void CardHouse.StackLayout.ApplySpacing (
            List< Card > cards,
            SeekerSetList seekerSets = null )  [protected], [virtual]
```

Implements CardHouse.CardGroupSettings.

Definition at line 12 of file StackLayout.cs.

6.134.3 Member Data Documentation

6.134.3.1 MarginalCardOffset

```
Vector3 CardHouse.StackLayout.MarginalCardOffset = new Vector3(0.01f, 0.01f, -0.01f)
```

Definition at line 8 of file StackLayout.cs.

6.134.3.2 SecondaryCollider

```
TriggerEnterRelay CardHouse.StackLayout.SecondaryCollider
```

Definition at line 9 of file StackLayout.cs.

6.134.3.3 Straighten

```
bool CardHouse.StackLayout.Straighten = true
```

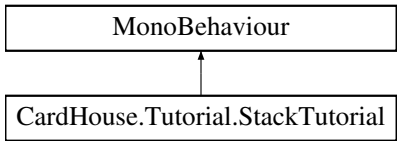
Definition at line 10 of file StackLayout.cs.

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

- StackLayout.cs

6.135 CardHouse.Tutorial.StackTutorial Class Reference

Inheritance diagram for CardHouse.Tutorial.StackTutorial:



Public Member Functions

- void AdjustXOffset ()
- void AdjustYOffset ()
- void UseColumnPreset ()
- void UseDeckPreset ()
- void UseCompactDeckPreset ()
- void UseRowPreset ()

Public Attributes

- Slider XOffsetSlider
- TMP_Text XOffsetText
- Slider YOffsetSlider
- TMP_Text YOffsetText
- StackLayout Stack

6.135.1 Detailed Description

Definition at line 7 of file StackTutorial.cs.

6.135.2 Member Function Documentation

6.135.2.1 AdjustXOffset()

```
void CardHouse.Tutorial.StackTutorial.AdjustXOffset ( )
```

Definition at line 16 of file StackTutorial.cs.

6.135.2.2 AdjustYOffset()

```
void CardHouse.Tutorial.StackTutorial.AdjustYOffset ( )
```

Definition at line 21 of file StackTutorial.cs.

6.135.2.3 UseColumnPreset()

```
void CardHouse.Tutorial.StackTutorial.UseColumnPreset ( )
```

Definition at line 41 of file StackTutorial.cs.

6.135.2.4 UseCompactDeckPreset()

```
void CardHouse.Tutorial.StackTutorial.UseCompactDeckPreset ( )
```

Definition at line 53 of file StackTutorial.cs.

6.135.2.5 UseDeckPreset()

```
void CardHouse.Tutorial.StackTutorial.UseDeckPreset ( )
```

Definition at line 47 of file StackTutorial.cs.

6.135.2.6 UseRowPreset()

```
void CardHouse.Tutorial.StackTutorial.UseRowPreset ( )
```

Definition at line 59 of file StackTutorial.cs.

6.135.3 Member Data Documentation

6.135.3.1 Stack

```
StackLayout CardHouse.Tutorial.StackTutorial.Stack
```

Definition at line 14 of file StackTutorial.cs.

6.135.3.2 XOffsetSlider

```
Slider CardHouse.Tutorial.StackTutorial.XOffsetSlider
```

Definition at line 9 of file StackTutorial.cs.

6.135.3.3 XOffsetText

```
TMP_Text CardHouse.Tutorial.StackTutorial.XOffsetText
```

Definition at line 10 of file StackTutorial.cs.

6.135.3.4 YOffsetSlider

```
Slider CardHouse.Tutorial.StackTutorial.YOffsetSlider
```

Definition at line 11 of file StackTutorial.cs.

6.135.3.5 YOffsetText

```
TMP_Text CardHouse.Tutorial.StackTutorial.YOffsetText
```

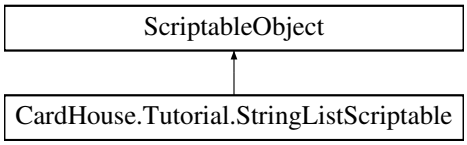
Definition at line 12 of file StackTutorial.cs.

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

- StackTutorial.cs

6.136 CardHouse.Tutorial.StringListScriptable Class Reference

Inheritance diagram for CardHouse.Tutorial.StringListScriptable:



Public Attributes

- List< string > MyList

6.136.1 Detailed Description

Definition at line 6 of file StringListScriptable.cs.

6.136.2 Member Data Documentation

6.136.2.1 MyList

```
List<string> CardHouse.Tutorial.StringListScriptable.MyList
```

Definition at line 8 of file StringListScriptable.cs.

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

- StringListScriptable.cs

6.137 CardHouse.Tutorial.StringSeekerKVP Class Reference

Public Attributes

- string Key
- ScriptableObject Value

6.137.1 Detailed Description

Definition at line 62 of file SeekerTutorial.cs.

6.137.2 Member Data Documentation

6.137.2.1 Key

```
string CardHouse.Tutorial.StringSeekerKVP.Key
```

Definition at line 64 of file SeekerTutorial.cs.

6.137.2.2 Value

```
ScriptableObject CardHouse.Tutorial.StringSeekerKVP.Value
```

Definition at line 65 of file SeekerTutorial.cs.

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

- SeekerTutorial.cs

6.138 CardHouse.StringUnityActionKvp Class Reference

Public Attributes

- string Key
- UnityEvent Value

6.138.1 Detailed Description

Definition at line 25 of file PhaseConditional.cs.

6.138.2 Member Data Documentation

6.138.2.1 Key

```
string CardHouse.StringUnityActionKvp.Key
```

Definition at line 27 of file PhaseConditional.cs.

6.138.2.2 Value

```
UnityEvent CardHouse.StringUnityActionKvp.Value
```

Definition at line 28 of file PhaseConditional.cs.

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

- PhaseConditional.cs

6.139 CardHouse.TargetCardParams Class Reference

Public Attributes

- Card Source
- Card Target

6.139.1 Detailed Description

Definition at line 3 of file TargetCardParams.cs.

6.139.2 Member Data Documentation

6.139.2.1 Source

```
Card CardHouse.TargetCardParams.Source
```

Definition at line 5 of file TargetCardParams.cs.

6.139.2.2 Target

```
Card CardHouse.TargetCardParams.Target
```

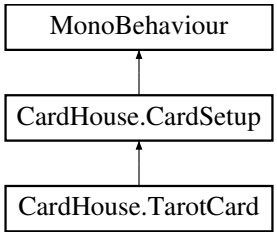
Definition at line 6 of file TargetCardParams.cs.

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

- TargetCardParams.cs

6.140 CardHouse.TarotCard Class Reference

Inheritance diagram for CardHouse.TarotCard:



Public Types

- enum **Arcana** { **Minor** , **Major** }

Public Member Functions

- override void Apply (CardDefinition cardDef)
- abstract void **Apply** (CardDefinition data)

Public Attributes

- SpriteRenderer Image

Properties

- ArcanaData ArcanaData `[get]`
- Arcana ArcanaType `[get]`

6.140.1 Detailed Description

Definition at line 5 of file TarotCard.cs.

6.140.2 Member Enumeration Documentation

6.140.2.1 Arcana

```
enum CardHouse.TarotCard.Arcana
```

Definition at line 7 of file TarotCard.cs.

6.140.3 Member Function Documentation

6.140.3.1 Apply()

```
override void CardHouse.TarotCard.Apply (
            CardDefinition cardDef )  [virtual]
```

Implements CardHouse.CardSetup.

Definition at line 18 of file TarotCard.cs.

6.140.4 Member Data Documentation

6.140.4.1 Image

```
SpriteRenderer CardHouse.TarotCard.Image
```

Definition at line 13 of file TarotCard.cs.

6.140.5 Property Documentation

6.140.5.1 ArcanaData

```
ArcanaData CardHouse.TarotCard.ArcanaData  [get]
```

Definition at line 15 of file TarotCard.cs.

6.140.5.2 ArcanaType

```
Arcana CardHouse.TarotCard.ArcanaType  [get]
```

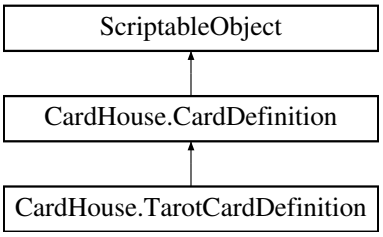
Definition at line 16 of file TarotCard.cs.

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

- TarotCard.cs

6.141 CardHouse.TarotCardDefinition Class Reference

Inheritance diagram for CardHouse.TarotCardDefinition:



Public Attributes

- ArcanaData Data
- Sprite Art

Public Attributes inherited from CardHouse.CardDefinition

- Sprite BackArt

6.141.1 Detailed Description

Definition at line 6 of file TarotCardDefinition.cs.

6.141.2 Member Data Documentation

6.141.2.1 Art

```
Sprite CardHouse.TarotCardDefinition.Art
```

Definition at line 9 of file TarotCardDefinition.cs.

6.141.2.2 Data

```
ArcanaData CardHouse.TarotCardDefinition.Data
```

Definition at line 8 of file TarotCardDefinition.cs.

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

- TarotCardDefinition.cs

6.142 CardHouse.SampleGames.Tarot.TarotSpread Class Reference

Public Member Functions

- void FillNext (Card card)

Public Attributes

- string Name
- string Instructions
- List< CardGroup > Slots

6.142.1 Detailed Description

Definition at line 8 of file TarotSpread.cs.

6.142.2 Member Function Documentation

6.142.2.1 FillNext()

```
void CardHouse.SampleGames.Tarot.TarotSpread.FillNext (
            Card card )
```

Definition at line 15 of file TarotSpread.cs.

6.142.3 Member Data Documentation

6.142.3.1 Instructions

```
string CardHouse.SampleGames.Tarot.TarotSpread.Instructions
```

Definition at line 12 of file TarotSpread.cs.

6.142.3.2 Name

```
string CardHouse.SampleGames.Tarot.TarotSpread.Name
```

Definition at line 10 of file TarotSpread.cs.

6.142.3.3 Slots

```
List<CardGroup> CardHouse.SampleGames.Tarot.TarotSpread.Slots
```

Definition at line 13 of file TarotSpread.cs.

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

- TarotSpread.cs

6.143 CardHouse.TimedEvent Class Reference

Public Member Functions

- IEnumerator ActivateAndDelay ()

Static Public Member Functions

- static IEnumerator ExecuteChain (List< TimedEvent > events, Action callback=null)

Public Attributes

- float Duration
- UnityEvent Event

6.143.1 Detailed Description

Definition at line 10 of file TimedEvent.cs.

6.143.2 Member Function Documentation

6.143.2.1 ActivateAndDelay()

```
IEnumerator CardHouse.TimedEvent.ActivateAndDelay ( )
```

Definition at line 15 of file TimedEvent.cs.

6.143.2.2 ExecuteChain()

```
static IEnumerator CardHouse.TimedEvent.ExecuteChain (
            List< TimedEvent > events,
            Action callback = null )  [static]
```

Definition at line 21 of file TimedEvent.cs.

6.143.3 Member Data Documentation

6.143.3.1 Duration

```
float CardHouse.TimedEvent.Duration
```

Definition at line 12 of file TimedEvent.cs.

6.143.3.2 Event

```
UnityEvent CardHouse.TimedEvent.Event
```

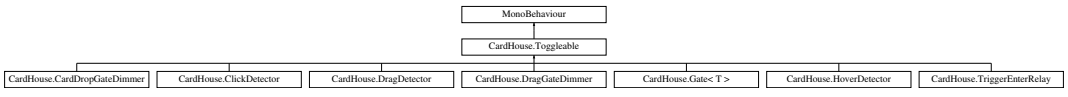
Definition at line 13 of file TimedEvent.cs.

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

- TimedEvent.cs

6.144 CardHouse.Toggleable Class Reference

Inheritance diagram for CardHouse.Toggleable:



Public Member Functions

- void SetIsActive (bool newValue)

Public Attributes

- bool IsActive = true

6.144.1 Detailed Description

Definition at line 5 of file Toggleable.cs.

6.144.2 Member Function Documentation

6.144.2.1 SetIsActive()

```
void CardHouse.Toggleable.SetIsActive (
            bool newValue )
```

Definition at line 9 of file Toggleable.cs.

6.144.3 Member Data Documentation

6.144.3.1 IsActive

```
bool CardHouse.Toggleable.IsActive = true
```

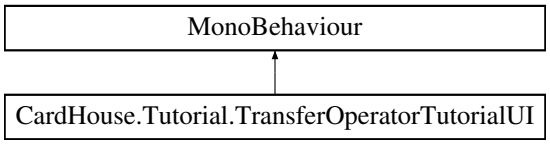
Definition at line 7 of file Toggleable.cs.

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

- Toggleable.cs

6.145 CardHouse.Tutorial.TransferOperatorTutorialUI Class Reference

Inheritance diagram for CardHouse.Tutorial.TransferOperatorTutorialUI:



Public Member Functions

- void AdjustNumberToTransfer ()
- void AdjustFlipSpeed ()
- void AdjustGrabFrom ()
- void AdjustSendTo ()

Public Attributes

- TMP_Dropdown GrabFromDropdown
- TMP_Dropdown SendToDropdown
- TMP_Text NumberToTransferText
- Slider NumberToTransferSlider
- TMP_Text FlipSpeedText
- Slider FlipSpeedSlider
- CardTransferOperator Operator

6.145.1 Detailed Description

Definition at line 7 of file TransferOperatorTutorialUI.cs.

6.145.2 Member Function Documentation

6.145.2.1 AdjustFlipSpeed()

```
void CardHouse.Tutorial.TransferOperatorTutorialUI.AdjustFlipSpeed ( )
```

Definition at line 23 of file TransferOperatorTutorialUI.cs.

6.145.2.2 AdjustGrabFrom()

```
void CardHouse.Tutorial.TransferOperatorTutorialUI.AdjustGrabFrom ( )
```

Definition at line 29 of file TransferOperatorTutorialUI.cs.

6.145.2.3 AdjustNumberToTransfer()

```
void CardHouse.Tutorial.TransferOperatorTutorialUI.AdjustNumberToTransfer ( )
```

Definition at line 17 of file TransferOperatorTutorialUI.cs.

6.145.2.4 AdjustSendTo()

```
void CardHouse.Tutorial.TransferOperatorTutorialUI.AdjustSendTo ( )
```

Definition at line 46 of file TransferOperatorTutorialUI.cs.

6.145.3 Member Data Documentation

6.145.3.1 FlipSpeedSlider

```
Slider CardHouse.Tutorial.TransferOperatorTutorialUI.FlipSpeedSlider
```

Definition at line 14 of file TransferOperatorTutorialUI.cs.

6.145.3.2 FlipSpeedText

```
TMP_Text CardHouse.Tutorial.TransferOperatorTutorialUI.FlipSpeedText
```

Definition at line 13 of file TransferOperatorTutorialUI.cs.

6.145.3.3 GrabFromDropdown

```
TMP_Dropdown CardHouse.Tutorial.TransferOperatorTutorialUI.GrabFromDropdown
```

Definition at line 9 of file TransferOperatorTutorialUI.cs.

6.145.3.4 NumberToTransferSlider

```
Slider CardHouse.Tutorial.TransferOperatorTutorialUI.NumberToTransferSlider
```

Definition at line 12 of file TransferOperatorTutorialUI.cs.

6.145.3.5 NumberToTransferText

```
TMP_Text CardHouse.Tutorial.TransferOperatorTutorialUI.NumberToTransferText
```

Definition at line 11 of file TransferOperatorTutorialUI.cs.

6.145.3.6 Operator

```
CardTransferOperator CardHouse.Tutorial.TransferOperatorTutorialUI.Operator
```

Definition at line 15 of file TransferOperatorTutorialUI.cs.

6.145.3.7 SendToDropdown

```
TMP_Dropdown CardHouse.Tutorial.TransferOperatorTutorialUI.SendToDropdown
```

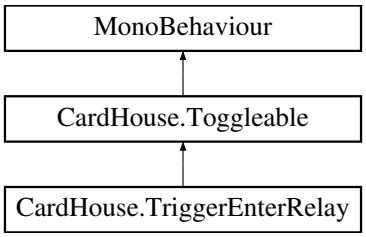
Definition at line 10 of file TransferOperatorTutorialUI.cs.

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

- TransferOperatorTutorialUI.cs

6.146 CardHouse.TriggerEnterRelay Class Reference

Inheritance diagram for CardHouse.TriggerEnterRelay:



Public Attributes

- CardGroup Relay

Public Attributes inherited from CardHouse.Toggleable

- bool IsActive = true

Additional Inherited Members

Public Member Functions inherited from CardHouse.Toggleable

- void SetIsActive (bool newValue)

6.146.1 Detailed Description

Definition at line 5 of file TriggerEnterRelay.cs.

6.146.2 Member Data Documentation

6.146.2.1 Relay

```
CardGroup CardHouse.TriggerEnterRelay.Relay
```

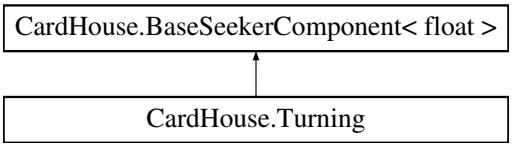
Definition at line 7 of file TriggerEnterRelay.cs.

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

- TriggerEnterRelay.cs

6.147 CardHouse.Turning Class Reference

Inheritance diagram for CardHouse.Turning:



Protected Member Functions

- override Seeker< float > GetDefaultSeeker ()
- override float GetCurrentValue ()
- override void SetNewValue (float value)

Protected Member Functions inherited from CardHouse.BaseSeekerComponent< float >

- abstract Seeker< T > **GetDefaultSeeker** ()
- abstract T **GetCurrentValue** ()
- abstract void **SetNewValue** (T value)

Additional Inherited Members

Public Member Functions inherited from CardHouse.BaseSeekerComponent< float >

- void StartSeeking (T destination, Seeker< T > strategy=null, bool useLocalSpace=false)

Public Attributes inherited from CardHouse.BaseSeekerComponent< float >

- SeekerScriptable< T > Strategy

Protected Attributes inherited from CardHouse.BaseSeekerComponent< float >

- Seeker< T > MyStrategy
- bool IsSeeking
- bool UseLocalSpace

6.147.1 Detailed Description

Definition at line 6 of file Turning.cs.

6.147.2 Member Function Documentation

6.147.2.1 GetCurrentValue()

```
override float CardHouse.Turning.GetCurrentValue ( )  [protected], [virtual]
```

Implements CardHouse.BaseSeekerComponent< float >.

Definition at line 13 of file Turning.cs.

6.147.2.2 GetDefaultSeeker()

```
override Seeker< float > CardHouse.Turning.GetDefaultSeeker ( )  [protected], [virtual]
```

Implements CardHouse.BaseSeekerComponent< float >.

Definition at line 8 of file Turning.cs.

6.147.2.3 SetNewValue()

```
override void CardHouse.Turning.SetNewValue (
            float value )  [protected]
```

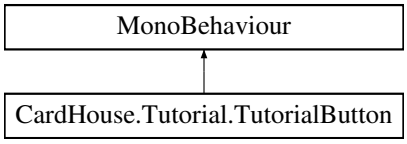
Definition at line 18 of file Turning.cs.

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

- Turning.cs

6.148 CardHouse.Tutorial.TutorialButton Class Reference

Inheritance diagram for CardHouse.Tutorial.TutorialButton:



Public Member Functions

- void Setup (string text, SandboxManager manager)

Public Attributes

- TMP_Text Label

6.148.1 Detailed Description

Definition at line 7 of file TutorialButton.cs.

6.148.2 Member Function Documentation

6.148.2.1 Setup()

```
void CardHouse.Tutorial.TutorialButton.Setup (
            string text,
            SandboxManager manager )
```

Definition at line 11 of file TutorialButton.cs.

6.148.3 Member Data Documentation

6.148.3.1 Label

```
TMP_Text CardHouse.Tutorial.TutorialButton.Label
```

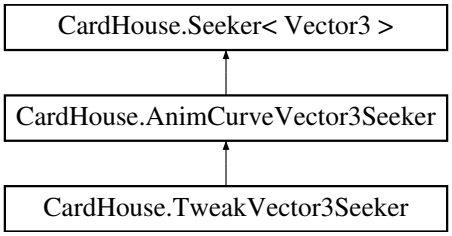
Definition at line 9 of file TutorialButton.cs.

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

- TutorialButton.cs

6.149 CardHouse.TweakVector3Seeker Class Reference

Inheritance diagram for CardHouse.TweakVector3Seeker:



Public Member Functions

- TweakVector3Seeker (float duration, AnimationCurve progressCurve, Vector3 tweak, AnimationCurve tweakMultiplier)
- override Seeker< Vector3 > MakeCopy ()
- override Vector3 Pump (Vector3 currentValue, float TimeSinceLastFrame)

Public Member Functions inherited from CardHouse.AnimCurveVector3Seeker

- AnimCurveVector3Seeker (float duration, AnimationCurve progressCurve)
- override Seeker< Vector3 > MakeCopy ()
- override Vector3 Pump (Vector3 currentValue, float TimeSinceLastFrame)
- override bool IsDone (Vector3 currentValue)

Public Member Functions inherited from CardHouse.Seeker< Vector3 >

- abstract Seeker< T > **MakeCopy** ()
- void StartSeeking (T from, T to)
- abstract T **Pump** (T currentValue, float TimeSinceLastFrame)
- abstract bool **IsDone** (T currentValue)

Public Attributes

- Vector3 Tweak
- AnimationCurve TweakMultiplier

Public Attributes inherited from CardHouse.AnimCurveVector3Seeker

- float Duration
- AnimationCurve ProgressCurve

Public Attributes inherited from CardHouse.Seeker< Vector3 >

- T End

Additional Inherited Members

Protected Attributes inherited from CardHouse.AnimCurveVector3Seeker

- float Timer

Protected Attributes inherited from CardHouse.Seeker< Vector3 >

- T Start

6.149.1 Detailed Description

Definition at line 5 of file TweakVector3Seeker.cs.

6.149.2 Constructor & Destructor Documentation

6.149.2.1 TweakVector3Seeker()

```
CardHouse.TweakVector3Seeker.TweakVector3Seeker (
            float duration,
            AnimationCurve progressCurve,
            Vector3 tweak,
            AnimationCurve tweakMultiplier )
```

Definition at line 10 of file TweakVector3Seeker.cs.

6.149.3 Member Function Documentation

6.149.3.1 MakeCopy()

```
override Seeker< Vector3 > CardHouse.TweakVector3Seeker.MakeCopy ( )  [virtual]
```

Reimplemented from CardHouse.AnimCurveVector3Seeker.

Definition at line 16 of file TweakVector3Seeker.cs.

6.149.3.2 Pump()

```
override Vector3 CardHouse.TweakVector3Seeker.Pump (
            Vector3 currentValue,
            float TimeSinceLastFrame )
```

Definition at line 21 of file TweakVector3Seeker.cs.

6.149.4 Member Data Documentation

6.149.4.1 Tweak

```
Vector3 CardHouse.TweakVector3Seeker.Tweak
```

Definition at line 7 of file TweakVector3Seeker.cs.

6.149.4.2 TweakMultiplier

```
AnimationCurve CardHouse.TweakVector3Seeker.TweakMultiplier
```

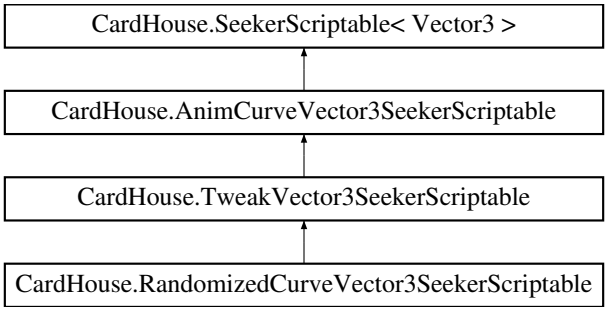
Definition at line 8 of file TweakVector3Seeker.cs.

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

- TweakVector3Seeker.cs

6.150 CardHouse.TweakVector3SeekerScriptable Class Reference

Inheritance diagram for CardHouse.TweakVector3SeekerScriptable:



Public Member Functions

- override Seeker< Vector3 > GetStrategy (params object[] args)
- override Seeker< Vector3 > GetStrategy (params object[] args)
- abstract Seeker< T > **GetStrategy** (params object[] args)

Public Attributes

- Vector3 Tweak
- AnimationCurve TweakMultiplier

Public Attributes inherited from CardHouse.AnimCurveVector3SeekerScriptable

- float Duration = 2f
- AnimationCurve ProgressCurve

6.150.1 Detailed Description

Definition at line 6 of file TweakVector3SeekerScriptable.cs.

6.150.2 Member Function Documentation

6.150.2.1 GetStrategy()

```
override Seeker< Vector3 > CardHouse.TweakVector3SeekerScriptable.GetStrategy (
            params object[] args )  [virtual]
```

Reimplemented from CardHouse.AnimCurveVector3SeekerScriptable.

Definition at line 11 of file TweakVector3SeekerScriptable.cs.

6.150.3 Member Data Documentation

6.150.3.1 Tweak

```
Vector3 CardHouse.TweakVector3SeekerScriptable.Tweak
```

Definition at line 8 of file TweakVector3SeekerScriptable.cs.

6.150.3.2 TweakMultiplier

```
AnimationCurve CardHouse.TweakVector3SeekerScriptable.TweakMultiplier
```

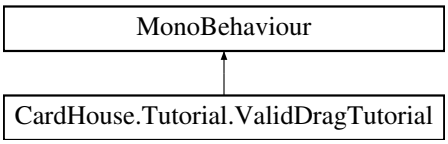
Definition at line 9 of file TweakVector3SeekerScriptable.cs.

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

- TweakVector3SeekerScriptable.cs

6.151 CardHouse.Tutorial.ValidDragTutorial Class Reference

Inheritance diagram for CardHouse.Tutorial.ValidDragTutorial:



Public Member Functions

- void UpdateDropdown10 ()
- void UpdateDropdown01 ()
- void UpdateDropdown11 ()
- void UpdateDropdown02 ()
- void UpdateDropdown12 ()

Public Attributes

- PhaseManager PhaseManager
- CardGroup GroupA
- CardGroup GroupB
- CardGroup GroupC
- CardGroup GroupD
- TMP_Dropdown Dropdown10
- TMP_Dropdown Dropdown01
- TMP_Dropdown Dropdown11
- TMP_Dropdown Dropdown02
- TMP_Dropdown Dropdown12

6.151.1 Detailed Description

Definition at line 6 of file ValidDragTutorial.cs.

6.151.2 Member Function Documentation

6.151.2.1 UpdateDropdown01()

```
void CardHouse.Tutorial.ValidDragTutorial.UpdateDropdown01 ( )
```

Definition at line 26 of file ValidDragTutorial.cs.

6.151.2.2 UpdateDropdown02()

```
void CardHouse.Tutorial.ValidDragTutorial.UpdateDropdown02 ( )
```

Definition at line 34 of file ValidDragTutorial.cs.

6.151.2.3 UpdateDropdown10()

```
void CardHouse.Tutorial.ValidDragTutorial.UpdateDropdown10 ( )
```

Definition at line 21 of file ValidDragTutorial.cs.

6.151.2.4 UpdateDropdown11()

```
void CardHouse.Tutorial.ValidDragTutorial.UpdateDropdown11 ( )
```

Definition at line 30 of file ValidDragTutorial.cs.

6.151.2.5 UpdateDropdown12()

```
void CardHouse.Tutorial.ValidDragTutorial.UpdateDropdown12 ( )
```

Definition at line 38 of file ValidDragTutorial.cs.

6.151.3 Member Data Documentation

6.151.3.1 Dropdown01

```
TMP_Dropdown CardHouse.Tutorial.ValidDragTutorial.Dropdown01
```

Definition at line 16 of file ValidDragTutorial.cs.

6.151.3.2 Dropdown02

```
TMP_Dropdown CardHouse.Tutorial.ValidDragTutorial.Dropdown02
```

Definition at line 18 of file ValidDragTutorial.cs.

6.151.3.3 Dropdown10

```
TMP_Dropdown CardHouse.Tutorial.ValidDragTutorial.Dropdown10
```

Definition at line 15 of file ValidDragTutorial.cs.

6.151.3.4 Dropdown11

```
TMP_Dropdown CardHouse.Tutorial.ValidDragTutorial.Dropdown11
```

Definition at line 17 of file ValidDragTutorial.cs.

6.151.3.5 Dropdown12

```
TMP_Dropdown CardHouse.Tutorial.ValidDragTutorial.Dropdown12
```

Definition at line 19 of file ValidDragTutorial.cs.

6.151.3.6 GroupA

```
CardGroup CardHouse.Tutorial.ValidDragTutorial.GroupA
```

Definition at line 10 of file ValidDragTutorial.cs.

6.151.3.7 GroupB

```
CardGroup CardHouse.Tutorial.ValidDragTutorial.GroupB
```

Definition at line 11 of file ValidDragTutorial.cs.

6.151.3.8 GroupC

```
CardGroup CardHouse.Tutorial.ValidDragTutorial.GroupC
```

Definition at line 12 of file ValidDragTutorial.cs.

6.151.3.9 GroupD

```
CardGroup CardHouse.Tutorial.ValidDragTutorial.GroupD
```

Definition at line 13 of file ValidDragTutorial.cs.

6.151.3.10 PhaseManager

```
PhaseManager CardHouse.Tutorial.ValidDragTutorial.PhaseManager
```

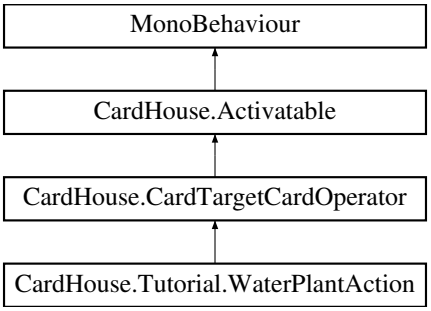
Definition at line 8 of file ValidDragTutorial.cs.

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

- ValidDragTutorial.cs

6.152 CardHouse.Tutorial.WaterPlantAction Class Reference

Inheritance diagram for CardHouse.Tutorial.WaterPlantAction:



Protected Member Functions

- override void ActOnTarget ()

Protected Member Functions inherited from CardHouse.CardTargetCardOperator

- override void OnActivate ()
- abstract void **ActOnTarget** ()

- virtual void OnActivate ()

Additional Inherited Members

Public Member Functions inherited from CardHouse.Activatable

- void Activate ()

Public Attributes inherited from CardHouse.CardTargetCardOperator

- SeekerScriptableSet DiscardSeekers

Protected Attributes inherited from CardHouse.CardTargetCardOperator

- Card MyCard
- Card Target

6.152.1 Detailed Description

Definition at line 3 of file WaterPlantAction.cs.

6.152.2 Member Function Documentation

6.152.2.1 ActOnTarget()

```
override void CardHouse.Tutorial.WaterPlantAction.ActOnTarget ( )  [protected], [virtual]
```

Implements CardHouse.CardTargetCardOperator.

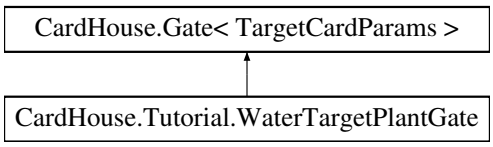
Definition at line 5 of file WaterPlantAction.cs.

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

- WaterPlantAction.cs

6.153 CardHouse.Tutorial.WaterTargetPlantGate Class Reference

Inheritance diagram for CardHouse.Tutorial.WaterTargetPlantGate:



Protected Member Functions

- override bool IsUnlockedInternal (TargetCardParams gateParams)

Protected Member Functions inherited from CardHouse.Gate< TargetCardParams >

- abstract bool **IsUnlockedInternal** (T argObject)

Additional Inherited Members

Public Member Functions inherited from CardHouse.Gate< TargetCardParams >

- bool IsUnlocked (T argObject)

6.153.1 Detailed Description

Definition at line 3 of file WaterTargetPlantGate.cs.

6.153.2 Member Function Documentation

6.153.2.1 IsUnlockedInternal()

```
override bool CardHouse.Tutorial.WaterTargetPlantGate.IsUnlockedInternal (
            TargetCardParams gateParams )  [protected]
```

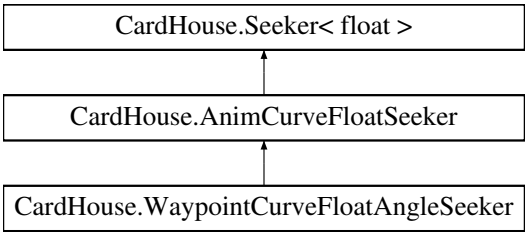
Definition at line 5 of file WaterTargetPlantGate.cs.

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

- WaterTargetPlantGate.cs

6.154 CardHouse.WaypointCurveFloatAngleSeeker Class Reference

Inheritance diagram for CardHouse.WaypointCurveFloatAngleSeeker:



Public Member Functions

- WaypointCurveFloatAngleSeeker (float duration, AnimationCurve progressCurve, List< float > waypoints)
- override Seeker< float > MakeCopy ()
- override float Pump (float currentValue, float TimeSinceLastFrame)
- override bool IsDone (float currentValue)

Public Member Functions inherited from CardHouse.AnimCurveFloatSeeker

- AnimCurveFloatSeeker (float duration, AnimationCurve progressCurve)
- override Seeker< float > MakeCopy ()
- override float Pump (float currentValue, float TimeSinceLastFrame)
- override bool IsDone (float currentValue)

Public Member Functions inherited from CardHouse.Seeker< float >

- abstract Seeker< T > **MakeCopy** ()
- void StartSeeking (T from, T to)
- abstract T **Pump** (T currentValue, float TimeSinceLastFrame)
- abstract bool **IsDone** (T currentValue)

Additional Inherited Members

Public Attributes inherited from CardHouse.AnimCurveFloatSeeker

- float Duration
- AnimationCurve ProgressCurve

Public Attributes inherited from CardHouse.Seeker< float >

- T End

Protected Attributes inherited from CardHouse.AnimCurveFloatSeeker

- float Timer

Protected Attributes inherited from CardHouse.Seeker< float >

- T Start

6.154.1 Detailed Description

Definition at line 7 of file WaypointCurveFloatAngleSeeker.cs.

6.154.2 Constructor & Destructor Documentation

6.154.2.1 WaypointCurveFloatAngleSeeker()

```
CardHouse.WaypointCurveFloatAngleSeeker.WaypointCurveFloatAngleSeeker (
            float duration,
            AnimationCurve progressCurve,
            List< float > waypoints )
```

Definition at line 11 of file WaypointCurveFloatAngleSeeker.cs.

6.154.3 Member Function Documentation

6.154.3.1 IsDone()

```
override bool CardHouse.WaypointCurveFloatAngleSeeker.IsDone (
            float currentValue )
```

Definition at line 38 of file WaypointCurveFloatAngleSeeker.cs.

6.154.3.2 MakeCopy()

```
override Seeker< float > CardHouse.WaypointCurveFloatAngleSeeker.MakeCopy ( )  [virtual]
```

Reimplemented from CardHouse.AnimCurveFloatSeeker.

Definition at line 22 of file WaypointCurveFloatAngleSeeker.cs.

6.154.3.3 Pump()

```
override float CardHouse.WaypointCurveFloatAngleSeeker.Pump (
            float currentValue,
            float TimeSinceLastFrame )
```

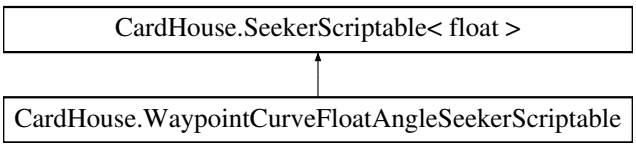
Definition at line 27 of file WaypointCurveFloatAngleSeeker.cs.

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

- WaypointCurveFloatAngleSeeker.cs

6.155 CardHouse.WaypointCurveFloatAngleSeekerScriptable Class Reference

Inheritance diagram for CardHouse.WaypointCurveFloatAngleSeekerScriptable:



Public Member Functions

- override Seeker< float > GetStrategy (params object[ ] args)
- abstract Seeker< T > **GetStrategy** (params object[ ] args)

Public Attributes

- float Duration = 2f
- AnimationCurve ProgressCurve

6.155.1 Detailed Description

Definition at line 7 of file WaypointCurveFloatAngleSeekerScriptable.cs.

6.155.2 Member Function Documentation

6.155.2.1 GetStrategy()

```
override Seeker< float > CardHouse.WaypointCurveFloatAngleSeekerScriptable.GetStrategy (
            params object[] args )  [virtual]
```

Implements CardHouse.SeekerScriptable< float >.

Definition at line 12 of file WaypointCurveFloatAngleSeekerScriptable.cs.

6.155.3 Member Data Documentation

6.155.3.1 Duration

```
float CardHouse.WaypointCurveFloatAngleSeekerScriptable.Duration = 2f
```

Definition at line 9 of file WaypointCurveFloatAngleSeekerScriptable.cs.

6.155.3.2 ProgressCurve

```
AnimationCurve CardHouse.WaypointCurveFloatAngleSeekerScriptable.ProgressCurve
```

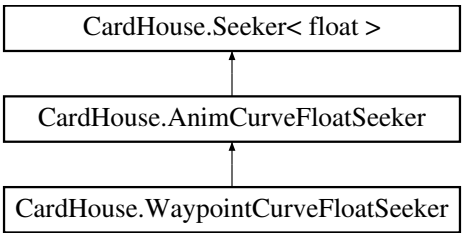
Definition at line 10 of file WaypointCurveFloatAngleSeekerScriptable.cs.

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

- WaypointCurveFloatAngleSeekerScriptable.cs

6.156 CardHouse.WaypointCurveFloatSeeker Class Reference

Inheritance diagram for CardHouse.WaypointCurveFloatSeeker:



Public Member Functions

- WaypointCurveFloatSeeker (float duration, AnimationCurve progressCurve, List< float > waypoints)
- override Seeker< float > MakeCopy ()
- override float Pump (float currentValue, float TimeSinceLastFrame)
- override bool IsDone (float currentValue)

Public Member Functions inherited from CardHouse.AnimCurveFloatSeeker

- AnimCurveFloatSeeker (float duration, AnimationCurve progressCurve)
- override Seeker< float > MakeCopy ()
- override float Pump (float currentValue, float TimeSinceLastFrame)
- override bool IsDone (float currentValue)

Public Member Functions inherited from CardHouse.Seeker< float >

- abstract Seeker< T > **MakeCopy** ()
- void StartSeeking (T from, T to)
- abstract T **Pump** (T currentValue, float TimeSinceLastFrame)
- abstract bool **IsDone** (T currentValue)

Additional Inherited Members

Public Attributes inherited from CardHouse.AnimCurveFloatSeeker

- float Duration
- AnimationCurve ProgressCurve

Public Attributes inherited from CardHouse.Seeker< float >

- T End

Protected Attributes inherited from CardHouse.AnimCurveFloatSeeker

- float Timer

Protected Attributes inherited from CardHouse.Seeker< float >

- T Start

6.156.1 Detailed Description

Definition at line 7 of file WaypointCurveFloatSeeker.cs.

6.156.2 Constructor & Destructor Documentation

6.156.2.1 WaypointCurveFloatSeeker()

```
CardHouse.WaypointCurveFloatSeeker.WaypointCurveFloatSeeker (
            float duration,
            AnimationCurve progressCurve,
            List< float > waypoints )
```

Definition at line 11 of file WaypointCurveFloatSeeker.cs.

6.156.3 Member Function Documentation

6.156.3.1 IsDone()

```
override bool CardHouse.WaypointCurveFloatSeeker.IsDone (
            float currentValue )
```

Definition at line 32 of file WaypointCurveFloatSeeker.cs.

6.156.3.2 MakeCopy()

```
override Seeker< float > CardHouse.WaypointCurveFloatSeeker.MakeCopy ( )  [virtual]
```

Reimplemented from CardHouse.AnimCurveFloatSeeker.

Definition at line 16 of file WaypointCurveFloatSeeker.cs.

6.156.3.3 Pump()

```
override float CardHouse.WaypointCurveFloatSeeker.Pump (
            float currentValue,
            float TimeSinceLastFrame )
```

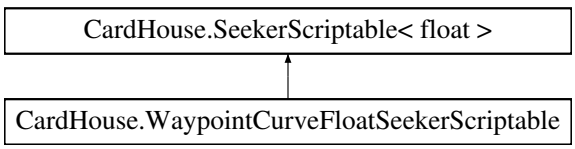
Definition at line 21 of file WaypointCurveFloatSeeker.cs.

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

- WaypointCurveFloatSeeker.cs

6.157 CardHouse.WaypointCurveFloatSeekerScriptable Class Reference

Inheritance diagram for CardHouse.WaypointCurveFloatSeekerScriptable:



Public Member Functions

- override Seeker< float > GetStrategy (params object[ ] args)
- abstract Seeker< T > **GetStrategy** (params object[ ] args)

Public Attributes

- float Duration = 2f
- AnimationCurve ProgressCurve

6.157.1 Detailed Description

Definition at line 7 of file WaypointCurveFloatSeekerScriptable.cs.

6.157.2 Member Function Documentation

6.157.2.1 GetStrategy()

```
override Seeker< float > CardHouse.WaypointCurveFloatSeekerScriptable.GetStrategy (
            params object[] args )  [virtual]
```

Implements CardHouse.SeekerScriptable< float >.

Definition at line 12 of file WaypointCurveFloatSeekerScriptable.cs.

6.157.3 Member Data Documentation

6.157.3.1 Duration

```
float CardHouse.WaypointCurveFloatSeekerScriptable.Duration = 2f
```

Definition at line 9 of file WaypointCurveFloatSeekerScriptable.cs.

6.157.3.2 ProgressCurve

```
AnimationCurve CardHouse.WaypointCurveFloatSeekerScriptable.ProgressCurve
```

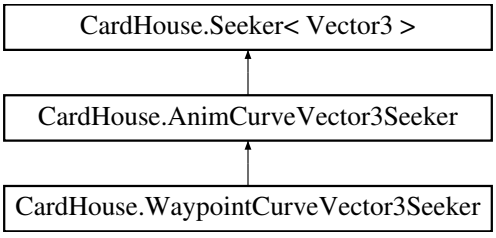
Definition at line 10 of file WaypointCurveFloatSeekerScriptable.cs.

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

- WaypointCurveFloatSeekerScriptable.cs

6.158 CardHouse.WaypointCurveVector3Seeker Class Reference

Inheritance diagram for CardHouse.WaypointCurveVector3Seeker:



Public Member Functions

- WaypointCurveVector3Seeker (float duration, AnimationCurve progressCurve, List< Vector3 > waypoints)
- override Seeker< Vector3 > MakeCopy ()
- override Vector3 Pump (Vector3 currentValue, float TimeSinceLastFrame)
- override bool IsDone (Vector3 currentValue)

Public Member Functions inherited from CardHouse.AnimCurveVector3Seeker

- AnimCurveVector3Seeker (float duration, AnimationCurve progressCurve)
- override Seeker< Vector3 > MakeCopy ()
- override Vector3 Pump (Vector3 currentValue, float TimeSinceLastFrame)
- override bool IsDone (Vector3 currentValue)

Public Member Functions inherited from CardHouse.Seeker< Vector3 >

- abstract Seeker< T > **MakeCopy** ()
- void StartSeeking (T from, T to)
- abstract T **Pump** (T currentValue, float TimeSinceLastFrame)
- abstract bool **IsDone** (T currentValue)

Additional Inherited Members

Public Attributes inherited from CardHouse.AnimCurveVector3Seeker

- float Duration
- AnimationCurve ProgressCurve

Public Attributes inherited from CardHouse.Seeker< Vector3 >

- T End

Protected Attributes inherited from CardHouse.AnimCurveVector3Seeker

- float Timer

Protected Attributes inherited from CardHouse.Seeker< Vector3 >

- T Start

6.158.1 Detailed Description

Definition at line 7 of file WaypointCurveVector3Seeker.cs.

6.158.2 Constructor & Destructor Documentation

6.158.2.1 WaypointCurveVector3Seeker()

```
CardHouse.WaypointCurveVector3Seeker.WaypointCurveVector3Seeker (
            float duration,
            AnimationCurve progressCurve,
            List< Vector3 > waypoints )
```

Definition at line 11 of file WaypointCurveVector3Seeker.cs.

6.158.3 Member Function Documentation

6.158.3.1 IsDone()

```
override bool CardHouse.WaypointCurveVector3Seeker.IsDone (
            Vector3 currentValue )
```

Definition at line 32 of file WaypointCurveVector3Seeker.cs.

6.158.3.2 MakeCopy()

```
override Seeker< Vector3 > CardHouse.WaypointCurveVector3Seeker.MakeCopy ( )  [virtual]
```

Reimplemented from CardHouse.AnimCurveVector3Seeker.

Definition at line 16 of file WaypointCurveVector3Seeker.cs.

6.158.3.3 Pump()

```
override Vector3 CardHouse.WaypointCurveVector3Seeker.Pump (
            Vector3 currentValue,
            float TimeSinceLastFrame )
```

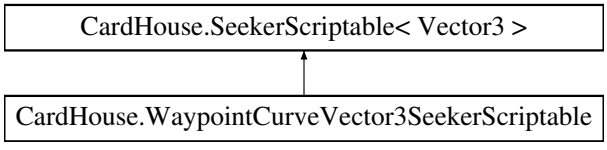
Definition at line 21 of file WaypointCurveVector3Seeker.cs.

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

- WaypointCurveVector3Seeker.cs

6.159 CardHouse.WaypointCurveVector3SeekerScriptable Class Reference

Inheritance diagram for CardHouse.WaypointCurveVector3SeekerScriptable:



Public Member Functions

- override Seeker< Vector3 > GetStrategy (params object[ ] args)
- abstract Seeker< T > **GetStrategy** (params object[ ] args)

Public Attributes

- float Duration = 2f
- AnimationCurve ProgressCurve

6.159.1 Detailed Description

Definition at line 7 of file WaypointCurveVector3SeekerScriptable.cs.

6.159.2 Member Function Documentation

6.159.2.1 GetStrategy()

```
override Seeker< Vector3 > CardHouse.WaypointCurveVector3SeekerScriptable.GetStrategy (
            params object[] args )  [virtual]
```

Implements CardHouse.SeekerScriptable< Vector3 >.

Definition at line 12 of file WaypointCurveVector3SeekerScriptable.cs.

6.159.3 Member Data Documentation

6.159.3.1 Duration

```
float CardHouse.WaypointCurveVector3SeekerScriptable.Duration = 2f
```

Definition at line 9 of file WaypointCurveVector3SeekerScriptable.cs.

6.159.3.2 ProgressCurve

```
AnimationCurve CardHouse.WaypointCurveVector3SeekerScriptable.ProgressCurve
```

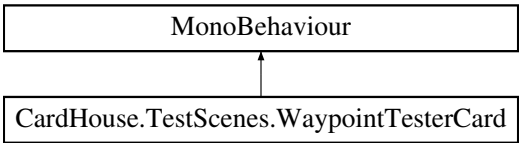
Definition at line 10 of file WaypointCurveVector3SeekerScriptable.cs.

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

- WaypointCurveVector3SeekerScriptable.cs

6.160 CardHouse.TestScenes.WaypointTesterCard Class Reference

Inheritance diagram for CardHouse.TestScenes.WaypointTesterCard:



Public Member Functions

- void Test ()

Public Attributes

- SeekerScriptableSet WaypointSeekers

6.160.1 Detailed Description

Definition at line 5 of file WaypointTesterCard.cs.

6.160.2 Member Function Documentation

6.160.2.1 Test()

```
void CardHouse.TestScenes.WaypointTesterCard.Test ( )
```

Definition at line 9 of file WaypointTesterCard.cs.

6.160.3 Member Data Documentation

6.160.3.1 WaypointSeekers

```
SeekerScriptableSet CardHouse.TestScenes.WaypointTesterCard.WaypointSeekers
```

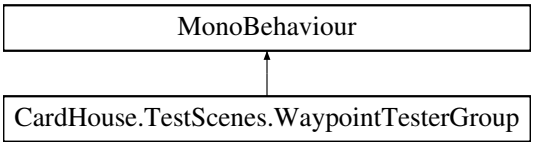
Definition at line 7 of file WaypointTesterCard.cs.

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

- WaypointTesterCard.cs

6.161 CardHouse.TestScenes.WaypointTesterGroup Class Reference

Inheritance diagram for CardHouse.TestScenes.WaypointTesterGroup:



Public Member Functions

- void Test (Card card, SeekerScriptable< Vector3 > homing, SeekerScriptable< float > turning, SeekerScriptable< float > scaling)

Public Attributes

- List< Transform > Waypoints

6.161.1 Detailed Description

Definition at line 8 of file WaypointTesterGroup.cs.

6.161.2 Member Function Documentation

6.161.2.1 Test()

```
void CardHouse.TestScenes.WaypointTesterGroup.Test (
            Card card,
            SeekerScriptable< Vector3 > homing,
            SeekerScriptable< float > turning,
            SeekerScriptable< float > scaling )
```

Definition at line 12 of file WaypointTesterGroup.cs.

6.161.3 Member Data Documentation

6.161.3.1 Waypoints

```
List<Transform> CardHouse.TestScenes.WaypointTesterGroup.Waypoints
```

Definition at line 10 of file WaypointTesterGroup.cs.

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

- WaypointTesterGroup.cs

Chapter 7

File Documentation

7.1 PokerCardDefinition.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     [CreateAssetMenu(menuName = "CardHouse/Card Definition/Poker")]
00006     public class PokerCardDefinition : CardDefinition
00007     {
00008         public int Rank;
00009         public PokerSuit Suit;
00010         public Sprite Art;
00011     }
00012 }
```

7.2 PokerCard.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     public class PokerCard : CardSetup
00006     {
00007         public SpriteRenderer Image;
00008         public SpriteRenderer BackImage;
00009         public PokerSuit Suit { get; private set; }
00010         public int Rank { get; private set; }
00011
00012         public override void Apply(CardDefinition data)
00013         {
00014             if (data is PokerCardDefinition pokerCard)
00015             {
00016                 Image.sprite = pokerCard.Art;
00017                 if (pokerCard.BackArt != null)
00018                 {
00019                     BackImage.sprite = pokerCard.BackArt;
00020                 }
00021                 Rank = pokerCard.Rank;
00022                 Suit = pokerCard.Suit;
00023             }
00024         }
00025     }
00026 }
```

7.3 PokerSuit.cs

```
00001 namespace CardHouse
00002 {
00003     public enum PokerSuit
00004     {
00005         None,
```

```
00006         Hearts,
00007         Spades,
00008         Clubs,
00009         Diamonds
00010     }
00011 }
```

7.4 ArcanaData.cs

```
00001 using System;
00002
00003 namespace CardHouse
00004 {
00005     [Serializable]
00006     public class ArcanaData
00007     {
00008         public MajorArcanaName Arcana;
00009         public TarotSuit Suit;
00010         public int Rank;
00011     }
00012 }
```

7.5 MajorArcanaName.cs

```
00001 namespace CardHouse
00002 {
00003     public enum MajorArcanaName
00004     {
00005         None,
00006         Fool,
00007         Magician,
00008         HighPriestess,
00009         Empress,
00010         Emperor,
00011         Hierophant,
00012         Lovers,
00013         Chariot,
00014         Strength,
00015         Hermit,
00016         WheelOfFortune,
00017         Justice,
00018         HangedMan,
00019         Death,
00020         Temperance,
00021         Devil,
00022         Tower,
00023         Star,
00024         Moon,
00025         Judgement,
00026         World,
00027         Sun
00028     }
00029 }
```

7.6 TarotCard.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     public class TarotCard : CardSetup
00006     {
00007         public enum Arcana
00008         {
00009             Minor,
00010             Major
00011         }
00012
00013         public SpriteRenderer Image;
00014
00015         public ArcanaData ArcanaData { get; private set; }
00016         public Arcana ArcanaType { get { return ArcanaData?.Arcana == null ? Arcana.Minor :
      Arcana.Major; } }
00017
00018         public override void Apply(CardDefinition cardDef)
```

```
00019         {
00020             if (cardDef is TarotCardDefinition tarotCard)
00021             {
00022                 ArcanaData = tarotCard.Data;
00023                 Image.sprite = tarotCard.Art;
00024             }
00025         }
00026     }
00027 }
```

7.7 TarotCardDefinition.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     [CreateAssetMenu(menuName = "CardHouse/Card Definition/Tarot")]
00006     public class TarotCardDefinition : CardDefinition
00007     {
00008         public ArcanaData Data;
00009         public Sprite Art;
00010     }
00011 }
```

7.8 TarotSuit.cs

```
00001 namespace CardHouse
00002 {
00003     public enum TarotSuit
00004     {
00005         None,
00006         Swords,
00007         Cups,
00008         Pentacles,
00009         Wands
00010     }
00011 }
```

7.9 Card.cs

```
00001 using System;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 using UnityEngine.Events;
00005
00006 namespace CardHouse
00007 {
00008     [RequireComponent(typeof(Homing)), RequireComponent(typeof(Turning)),
      RequireComponent(typeof(Scaling))]
00009     public class Card : MonoBehaviour
00010     {
00011         [Serializable]
00012         public class GroupTransitionEvent
00013         {
00014             public GroupName Group;
00015             public UnityEvent EntryEvent;
00016             public UnityEvent ExitEvent;
00017         }
00018
00019         [HideInInspector]
00020         public CardGroup Group;
00021         public Homing Homing { get; private set; }
00022         public Turning Turning { get; private set; }
00023         public Scaling Scaling { get; private set; }
00024
00025         public Animator FlipAnimator;
00026
00027         public bool CanBeUpsideDown;
00028         [Range(0f, 1f)]
00029         public float UpsideDownChance = 0.5f;
00030         public Transform RootToRotateWhenUpsideDown;
00031
00032         public Homing FaceHoming;
00033         public Turning FaceTurning;
00034         public Scaling FaceScaling;
```

```
00035
00036         public List<GroupTransitionEvent> GroupTransitionEvents;
00037
00038         public CardFacing Facing { get { return FlipAnimator.GetBool("FaceUp") ? CardFacing.FaceUp :
      CardFacing.FaceDown; } }
00039
00040         public UnityEvent OnFlipUp;
00041         public UnityEvent OnFlipDown;
00042         public UnityEvent OnPlay;
00043
00044         public Action<Card, CardGroup> OnMount;
00045
00046         bool IsFocused;
00047
00048         public static Action<Card> OnCardFocused;
00049
00050         void Awake()
00051         {
00052             Homing = GetComponent<Homing>();
00053             Turning = GetComponent<Turning>();
00054             Scaling = GetComponent<Scaling>();
00055             OnCardFocused += HandleCardFocused;
00056         }
00057
00058         void OnDestroy()
00059         {
00060             OnCardFocused -= HandleCardFocused;
00061         }
00062
00063         void Update()
00064         {
00065             if (IsFocused && Input.GetMouseButtonDown(0))
00066             {
00067                 SetFocus(false);
00068             }
00069         }
00070
00071         public void SetFacing(bool isFaceUp)
00072         {
00073             SetFacing(isFaceUp ? CardFacing.FaceUp : CardFacing.FaceDown);
00074         }
00075
00076         public void SetFacing(CardFacing facing, bool immediate = false, float spd = 1f)
00077         {
00078             if (facing == CardFacing.None)
00079                 return;
00080
00081             FlipAnimator.SetBool("SkipAnimation", immediate);
00082             FlipAnimator.SetBool("FaceUp", facing == CardFacing.FaceUp);
00083             FlipAnimator.speed = spd;
00084
00085             if (facing == CardFacing.FaceUp)
00086             {
00087                 OnFlipUp?.Invoke();
00088             }
00089             else if (facing == CardFacing.FaceDown)
00090             {
00091                 OnFlipDown?.Invoke();
00092             }
00093         }
00094
00095         public void SetUpsideDown(bool isUpsideDown)
00096         {
00097             if (!CanBeUpsideDown)
00098                 return;
00099
00100             var currentRotation = RootToRotateWhenUpsideDown.localRotation.eulerAngles;
00101             currentRotation += Vector3.forward * ((isUpsideDown ? 180f : 0f) -
      RootToRotateWhenUpsideDown.localRotation.eulerAngles.z);
00102
00103             RootToRotateWhenUpsideDown.localRotation = Quaternion.Euler(currentRotation);
00104         }
00105
00106         public bool IsUpsideDown => CanBeUpsideDown &&
      (Mathf.Abs(RootToRotateWhenUpsideDown.localRotation.eulerAngles.z) - 180f) < 1f;
00107
00108         public void HandlePlayed()
00109         {
00110             OnPlay.Invoke();
00111         }
00112
00113         public CardGroup GetDiscardGroup()
00114         {
00115             var ownerIndex = GroupRegistry.Instance.GetOwnerIndex(Group);
00116             if (ownerIndex == null && GetComponent<CardLoyalty>() != null)
00117             {
00118                 ownerIndex = GetComponent<CardLoyalty>().PlayerIndex;
```

```
00119             }
00120             var discardGroup = GroupRegistry.Instance?.Get(GroupName.Discard, ownerIndex);
00121             return discardGroup;
00122         }
00123 
00124         public void SetFocus(bool isFocused)
00125         {
00126             IsFocused = isFocused;
00127             FaceHoming.StartSeeking(isFocused ? Camera.main.transform.position + Vector3.forward * 2f
      : Vector3.zero, useLocalSpace: !isFocused);
00128             FaceTurning.StartSeeking(isFocused ? Camera.main.transform.rotation.eulerAngles.z : 0,
      useLocalSpace: !isFocused);
00129             FaceScaling.StartSeeking(isFocused ? 2f * Camera.main.orthographicSize / 4f : 1f,
      useLocalSpace: !isFocused);
00130             if (isFocused)
00131             {
00132                 OnCardFocused?.Invoke(this);
00133             }
00134         }
00135 
00136         void HandleCardFocused(Card card)
00137         {
00138             if (IsFocused && card != this)
00139             {
00140                 SetFocus(false);
00141             }
00142         }
00143 
00144         public void ToggleFocus()
00145         {
00146             SetFocus(!IsFocused);
00147         }
00148 
00149         public void TriggerMountEvents(CardGroup group)
00150         {
00151             OnMount?.Invoke(this, group);
00152 
00153             var groupName = GroupRegistry.Instance?.GetGroupName(group) ?? GroupName.None;
00154             foreach (var eventTransition in GroupTransitionEvents)
00155             {
00156                 if (eventTransition.Group == groupName)
00157                 {
00158                     eventTransition.EntryEvent?.Invoke();
00159                     break;
00160                 }
00161             }
00162         }
00163 
00164         public void TriggerUnMountEvents(GroupName group)
00165         {
00166             foreach (var eventTransition in GroupTransitionEvents)
00167             {
00168                 if (eventTransition.Group == group)
00169                 {
00170                     eventTransition.ExitEvent.Invoke();
00171                     break;
00172                 }
00173             }
00174         }
00175     }
00176 }
```

7.10 CardFacing.cs

```
00001 namespace CardHouse
00002 {
00003     public enum CardFacing
00004     {
00005         None,
00006         FaceUp,
00007         FaceDown
00008     }
00009 }
```

7.11 Activatable.cs

```
00001 using UnityEngine;
00002 
00003 namespace CardHouse
```

```
00004 {
00005     public class Activatable : MonoBehaviour
00006     {
00007         public void Activate()
00008         {
00009             OnActivate();
00010         }
00011
00012         protected virtual void OnActivate() { }
00013     }
00014 }
```

7.12 CardTargetCardOperator.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     [RequireComponent(typeof(Card))]
00006     public abstract class CardTargetCardOperator : Activatable
00007     {
00008         public SeekerScriptableSet DiscardSeekers;
00009         protected Card MyCard;
00010         protected Card Target;
00011
00012         void Awake()
00013         {
00014             MyCard = GetComponent<Card>();
00015             CardGroup.OnCardUsedOnTarget += SetTarget;
00016         }
00017
00018         void OnDestroy()
00019         {
00020             CardGroup.OnCardUsedOnTarget -= SetTarget;
00021         }
00022
00023         void SetTarget(Card source, Card target)
00024         {
00025             if (source == MyCard)
00026             {
00027                 Target = target;
00028             }
00029         }
00030
00031         protected override void OnActivate()
00032         {
00033             ActOnTarget();
00034         }
00035
00036         protected abstract void ActOnTarget();
00037     }
00038 }
```

7.13 CardTransferOperator.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using UnityEngine;
00005
00006 namespace CardHouse
00007 {
00008     public class CardTransferOperator : Activatable
00009     {
00010         public GroupTransition Transition;
00011         public GroupTargetType GrabFrom = GroupTargetType.Last;
00012         public GroupTargetType SendTo = GroupTargetType.Last;
00013         public int NumberToTransfer = 1;
00014         public float FlipSpeed = 1;
00015
00016         public SeekerScriptable<Vector3> PopPushHomingOverride;
00017
00018         public List<TimedEvent> OnSourceDepletedEventChain;
00019
00020         public bool TryAgainAfterSourceDepleted;
00021
00022         protected override void OnActivate()
00023         {
```

```
00024             var cardsToMove = NumberToTransfer > 0 ? Transition.Source.Get(GrabFrom, NumberToTransfer)
      : Transition.Source.MountedCards.ToList();
00025
00026             TransferCards(cardsToMove);
00027
00028             if (NumberToTransfer > cardsToMove.Count)
00029             {
00030                 StartCoroutine(ExecuteOnSourceDepletedEventChain());
00031             }
00032         }
00033
00034         IEnumerator ExecuteOnSourceDepletedEventChain()
00035         {
00036             yield return TimedEvent.ExecuteChain(OnSourceDepletedEventChain);
00037             if (TryAgainAfterSourceDepleted)
00038             {
00039                 var cardsToMove = Transition.Source.Get(GrabFrom, NumberToTransfer);
00040                 TransferCards(cardsToMove);
00041             }
00042         }
00043
00044         void TransferCards(List<Card> cards)
00045         {
00046             foreach (var card in cards)
00047             {
00048                 Transition.Destination.Mount(card, SendTo == GroupTargetType.First ? -1 : SendTo ==
      GroupTargetType.Last ? null : UnityEngine.Random.Range(0, Transition.Destination.MountedCards.Count +
      1), seekerSets: new SeekerSetList { new SeekerSet { Card = card, Homing =
      PopPushHomingOverride?.GetStrategy(), FlipSpeed = FlipSpeed } });
00049             }
00050         }
00051     }
00052 }
```

7.14 DiscardCardOperator.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     public class DiscardCardOperator : MonoBehaviour
00006     {
00007         public void Activate()
00008         {
00009             var card = GetComponentInParent<Card>();
00010             var discardGroup = card.GetDiscardGroup();
00011             if (discardGroup != null)
00012             {
00013                 discardGroup.Mount(card);
00014             }
00015         }
00016     }
00017 }
```

7.15 DiscardTargetCardOperator.cs

```
00001 namespace CardHouse
00002 {
00003     public class DiscardTargetCardOperator : CardTargetCardOperator
00004     {
00005         public SeekerScriptableSet TargetDiscardSeekers;
00006
00007         protected override void ActOnTarget()
00008         {
00009             var discardGroup = Target.GetDiscardGroup();
00010             if (discardGroup != null)
00011             {
00012                 var seekerSets = new SeekerSetList { new SeekerSet { Card = Target, Homing =
      TargetDiscardSeekers.Homing?.GetStrategy() } };
00013                 var presentationTransform =
      PhaseManager.Instance?.CurrentPhase?.CardPresentationPosition;
00014                 if (presentationTransform != null)
00015                 {
00016                     seekerSets.Add(new SeekerSet
00017                     {
00018                         Card = MyCard,
00019                         Homing = DiscardSeekers.Homing.GetStrategy(presentationTransform.position),
00020                         Turning =
      DiscardSeekers.Turning.GetStrategy(CardHouse.Utils.CorrectAngle(presentationTransform.rotation.eulerAngles.z)),
```

```
00021                         Scaling =
      DiscardSeekers.Scaling.GetStrategy(presentationTransform.lossyScale.x)
00022                     });
00023                 }
00024                 discardGroup.Mount(Target,
00025                     seekerSets: seekerSets,
00026                     seekersForUnmounting: new SeekerSet { Homing =
      DiscardSeekers.Homing?.GetStrategy() }
00027                 );
00028             }
00029         }
00030     }
00031 }
```

7.16 ShuffleOperator.cs

```
00001 using System.Collections.Generic;
00002 using System.Linq;
00003 using UnityEngine.Serialization;
00004
00005 namespace CardHouse
00006 {
00007     public class ShuffleOperator : Activatable
00008     {
00009         [FormerlySerializedAs("Groups")]
00010         public List<CardGroup> GroupsToShuffleIntoDeck;
00011         public CardGroup Deck;
00012
00013         protected override void OnActivate()
00014         {
00015             foreach (var slot in GroupsToShuffleIntoDeck)
00016             {
00017                 foreach (var card in slot.MountedCards.ToList())
00018                 {
00019                     Deck.Mount(card);
00020                 }
00021             }
00022
00023             Deck.Shuffle();
00024         }
00025     }
00026 }
```

7.17 CardDefinition.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     public abstract class CardDefinition : ScriptableObject
00006     {
00007         public Sprite BackArt;
00008     }
00009 }
```

7.18 DeckDefinition.cs

```
00001 using System.Collections.Generic;
00002 using UnityEngine;
00003
00004 namespace CardHouse
00005 {
00006     [CreateAssetMenu(menuName = "CardHouse/Deck Definition")]
00007     public class DeckDefinition : ScriptableObject
00008     {
00009         public Sprite CardBackArt;
00010         public List<CardDefinition> CardCollection;
00011     }
00012 }
```

7.19 ClickDetector.cs

```
00001 using UnityEngine.Events;
00002
00003 namespace CardHouse
00004 {
00005     public class ClickDetector : Toggleable
00006     {
00007         public UnityEvent OnPress;
00008         public UnityEvent OnButtonClicked;
00009
00010         public GateCollection<NoParams> ClickGates;
00011
00012         void OnMouseDown()
00013         {
00014             if (IsActive && ClickGates.AllUnlocked(null))
00015             {
00016                 OnPress.Invoke();
00017             }
00018         }
00019
00020         void OnMouseUpAsButton()
00021         {
00022             if (IsActive && ClickGates.AllUnlocked(null))
00023             {
00024                 OnButtonClicked.Invoke();
00025             }
00026         }
00027     }
00028 }
```

7.20 PhaseGateClick.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     [RequireComponent(typeof(ClickDetector))]
00006     public class PhaseGateClick : Gate<NoParams>
00007     {
00008         ClickDetector MyClickDetector;
00009         private void Awake()
00010         {
00011             MyClickDetector = GetComponent<ClickDetector>();
00012         }
00013         protected override bool IsUnlockedInternal(NoParams gateParams)
00014         {
00015             return PhaseManager.Instance.IsValidClick(MyClickDetector);
00016         }
00017     }
00018 }
```

7.21 GroupConditional.cs

```
00001 using System;
00002 using System.Collections.Generic;
00003 using UnityEngine.Events;
00004
00005 namespace CardHouse
00006 {
00007     public class GroupConditional : Activatable
00008     {
00009         public Card MyCard;
00010         public List<GroupNameUnityActionKvp> Responses;
00011
00012         protected override void OnActivate()
00013         {
00014             foreach (var kvp in Responses)
00015             {
00016                 if (kvp.Key == GroupRegistry.Instance?.GetGroupName(MyCard.Group))
00017                 {
00018                     kvp.Value.Invoke();
00019                     break;
00020                 }
00021             }
00022         }
00023     }
00024
00025     [Serializable]
```

```
00026     public class GroupNameUnityActionKvp
00027     {
00028         public GroupName Key;
00029         public UnityEvent Value;
00030     }
00031 }
```

7.22 PhaseConditional.cs

```
00001 using System;
00002 using System.Collections.Generic;
00003 using UnityEngine.Events;
00004
00005 namespace CardHouse
00006 {
00007     public class PhaseConditional : Activatable
00008     {
00009         public List<StringUnityActionKvp> Responses;
00010
00011         protected override void OnActivate()
00012         {
00013             foreach (var kvp in Responses)
00014             {
00015                 if (kvp.Key == PhaseManager.Instance?.CurrentPhase.Name)
00016                 {
00017                     kvp.Value.Invoke();
00018                     break;
00019                 }
00020             }
00021         }
00022     }
00023
00024     [Serializable]
00025     public class StringUnityActionKvp
00026     {
00027         public string Key;
00028         public UnityEvent Value;
00029     }
00030 }
```

7.23 CurrencyChangeDetector.cs

```
00001 using UnityEngine;
00002 using UnityEngine.Events;
00003 using UnityEngine.Serialization;
00004
00005 namespace CardHouse
00006 {
00007     public class CurrencyChangeDetector : MonoBehaviour
00008     {
00009         [FormerlySerializedAs("OnResourceChange")]
00010         public UnityEvent OnCurrencyChange;
00011         CurrencyRegistry MyCurrencyRegistry;
00012
00013         void Start()
00014         {
00015             MyCurrencyRegistry = CurrencyRegistry.Instance;
00016             if (MyCurrencyRegistry != null)
00017             {
00018                 MyCurrencyRegistry.OnCurrencyChanged += HandleCurrencyChanged;
00019             }
00020         }
00021
00022         void OnDestroy()
00023         {
00024             if (MyCurrencyRegistry != null)
00025             {
00026                 MyCurrencyRegistry.OnCurrencyChanged -= HandleCurrencyChanged;
00027             }
00028         }
00029
00030         void HandleCurrencyChanged(int playerIndex, CurrencyWallet newResources)
00031         {
00032             OnCurrencyChange.Invoke();
00033         }
00034     }
00035 }
```

7.24 CurrencyContainer.cs

```
00001 using System;
00002
00003 namespace CardHouse
00004 {
00005     [Serializable]
00006     public class CurrencyContainer : CurrencyQuantity
00007     {
00008         public bool HasMax;
00009         public int Max;
00010         public bool HasMin = true;
00011         public int Min;
00012         public int RefillValue;
00013
00014         public void Adjust(int amount)
00015         {
00016             Amount += amount;
00017             if (HasMin && Amount < Min)
00018             {
00019                 Amount = Min;
00020             }
00021             if (HasMax && Amount > Max)
00022             {
00023                 Amount = Max;
00024             }
00025         }
00026
00027         public override object Clone()
00028         {
00029             return new CurrencyContainer { CurrencyType = CurrencyType, Amount = Amount, HasMax =
      HasMax, Max = Max, HasMin = HasMin, Min = Min, RefillValue = RefillValue };
00030         }
00031     }
00032 }
```

7.25 CurrencyCost.cs

```
00001 using System;
00002 using System.Collections.Generic;
00003 using TMPro;
00004 using UnityEngine;
00005
00006 namespace CardHouse
00007 {
00008     public class CurrencyCost : MonoBehaviour
00009     {
00010         [Serializable]
00011         public class CostWithLabel
00012         {
00013             public CurrencyQuantity Cost;
00014             public TextMeshPro Label;
00015         }
00016
00017         public List<CostWithLabel> Cost;
00018
00019         void Start()
00020         {
00021             foreach (var cost in Cost)
00022             {
00023                 if (cost.Label != null)
00024                 {
00025                     cost.Label.text = cost.Cost.Amount.ToString();
00026                 }
00027             }
00028         }
00029
00030         public void Activate()
00031         {
00032             foreach (var resource in Cost)
00033             {
00034                 CurrencyRegistry.Instance.AdjustCurrency(resource.Cost.CurrencyType.Name,
      PhaseManager.Instance.PlayerIndex, -1 * resource.Cost.Amount);
00035             }
00036         }
00037     }
00038 }
```

7.26 CurrencyQuantity.cs

```
00001 using System;
```

```
00002 using UnityEngine.Serialization;
00003
00004 namespace CardHouse
00005 {
00006     [Serializable]
00007     public class CurrencyQuantity : ICloneable
00008     {
00009         [FormerlySerializedAs("ResourceType")]
00010         public CurrencyScriptable CurrencyType;
00011         public int Amount;
00012
00013         public virtual object Clone()
00014         {
00015             return new CurrencyQuantity { CurrencyType = CurrencyType, Amount = Amount };
00016         }
00017     }
00018 }
```

7.27 CurrencyRegistry.cs

```
00001 using System;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 using UnityEngine.Serialization;
00005 using UnityEngine.UI;
00006
00007 namespace CardHouse
00008 {
00009     public class CurrencyRegistry : MonoBehaviour
00010     {
00011         public Text CurrentPlayerLabel;
00012
00013         [FormerlySerializedAs("ResourceDisplayParent")]
00014         public Transform CurrencyDisplayParent;
00015         [FormerlySerializedAs("ResourceDisplayPrefab")]
00016         public GameObject CurrencyDisplayPrefab;
00017
00018         [FormerlySerializedAs("PlayerCurrencies")]
00019         public List<CurrencyWallet> PlayerWallets;
00020
00021         public static CurrencyRegistry Instance;
00022
00023         PhaseManager MyPhaseManager;
00024
00025         Dictionary<CurrencyContainer, CurrencyUI> CurrencyUILookup = new Dictionary<CurrencyContainer,
      CurrencyUI>();
00026
00027         public Action<int, CurrencyWallet> OnCurrencyChanged;
00028
00029         void Awake()
00030         {
00031             Instance = this;
00032         }
00033
00034         void Start()
00035         {
00036             MyPhaseManager = PhaseManager.Instance;
00037             if (MyPhaseManager == null)
00038             {
00039                 Debug.LogError("Currency Registry requires a Phase Manager to work");
00040                 return;
00041             }
00042             MyPhaseManager.OnPhaseChanged += HandlePhaseChange;
00043
00044             if (PlayerWallets.Count > 0)
00045             {
00046                 ShowPlayerWallet(0);
00047             }
00048         }
00049
00050         void OnDestroy()
00051         {
00052             if (MyPhaseManager != null)
00053             {
00054                 MyPhaseManager.OnPhaseChanged -= HandlePhaseChange;
00055             }
00056         }
00057
00058         void HandlePhaseChange(Phase newPhase)
00059         {
00060             ShowPlayerWallet(PhaseManager.Instance.PlayerIndex);
00061         }
00062
```

```
00063         void ShowPlayerWallet(int playerIndex)
00064         {
00065             CurrentPlayerLabel.text = string.Format("Player {0}", PhaseManager.Instance.PlayerIndex +
      1);
00066 
00067             CurrencyUILookup.Clear();
00068             for (var i = 0; i < CurrencyDisplayParent.childCount; i++)
00069             {
00070                 Destroy(CurrencyDisplayParent.GetChild(i).gameObject);
00071             }
00072             foreach (var currency in PlayerWallets[playerIndex].Currencies)
00073             {
00074                 var newRow = Instantiate(CurrencyDisplayPrefab, CurrencyDisplayParent);
00075                 var currencyUI = newRow.GetComponent<CurrencyUI>();
00076                 CurrencyUILookup[currency] = currencyUI;
00077                 currencyUI.Apply(currency);
00078             }
00079         }
00080 
00081         public int? GetCurrency(string name, int playerIndex)
00082         {
00083             return FindCurrency(name, playerIndex)?.Amount;
00084         }
00085 
00086         public int? GetCurrency(CurrencyScriptable resourceDef, int playerIndex)
00087         {
00088             return FindCurrency(resourceDef.name, playerIndex)?.Amount;
00089         }
00090 
00091         CurrencyContainer FindCurrency(string name, int playerIndex)
00092         {
00093             if (playerIndex < 0 || playerIndex >= PlayerWallets.Count)
00094                 return null;
00095 
00096             return PlayerWallets[playerIndex].FindCurrency(name);
00097         }
00098 
00099         public void AdjustCurrency(string name, int playerIndex, int amount)
00100         {
00101             var currency = FindCurrency(name, playerIndex);
00102             if (currency != null)
00103             {
00104                 currency.Adjust(amount);
00105                 CurrencyUILookup[currency].Apply(currency);
00106                 OnCurrencyChanged?.Invoke(playerIndex, PlayerWallets[playerIndex]);
00107             }
00108         }
00109 
00110         public void Refill(string name, int playerIndex)
00111         {
00112             var currency = FindCurrency(name, playerIndex);
00113             if (currency != null)
00114             {
00115                 currency.Amount = currency.HasMax ? currency.Max : currency.Amount >
      currency.RefillValue ? currency.Amount : currency.RefillValue;
00116                 if (CurrencyUILookup.ContainsKey(currency))
00117                 {
00118                     CurrencyUILookup[currency].Apply(currency);
00119                 }
00120                 OnCurrencyChanged?.Invoke(playerIndex, PlayerWallets[playerIndex]);
00121             }
00122         }
00123     }
00124 }
```

7.28 CurrencyScriptable.cs

```
00001 using UnityEngine;
00002 
00003 namespace CardHouse
00004 {
00005     [CreateAssetMenu(menuName = "CardHouse/Currency")]
00006     public class CurrencyScriptable : ScriptableObject
00007     {
00008         public string Name;
00009         public Sprite Sprite;
00010     }
00011 }
```

7.29 CurrencyUI.cs

```
00001 using UnityEngine;
00002 using UnityEngine.UI;
00003
00004 namespace CardHouse
00005 {
00006     public class CurrencyUI : MonoBehaviour
00007     {
00008         public Image Image;
00009         public Text Text;
00010
00011         public void Apply(CurrencyContainer resource)
00012         {
00013             Image.sprite = resource.CurrencyType.Sprite;
00014             var text = resource.Amount.ToString();
00015             if (resource.HasMax)
00016             {
00017                 text += "/" + resource.Max.ToString();
00018             }
00019             Text.text = text;
00020         }
00021     }
00022 }
```

7.30 CurrencyWallet.cs

```
00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004
00005 namespace CardHouse
00006 {
00007     [Serializable]
00008     public class CurrencyWallet : ICloneable
00009     {
00010         public List<CurrencyContainer> Currencies;
00011
00012         public CurrencyContainer FindCurrency(string name)
00013         {
00014             foreach (var resource in Currencies)
00015             {
00016                 if (resource.CurrencyType.Name == name)
00017                 {
00018                     return resource;
00019                 }
00020             }
00021
00022             return null;
00023         }
00024
00025         public bool CanAfford(List<CurrencyQuantity> Cost)
00026         {
00027             foreach (var holder in Cost)
00028             {
00029                 var myHolder = FindCurrency(holder.CurrencyType.Name);
00030                 if (myHolder == null || myHolder.Amount < holder.Amount)
00031                 {
00032                     return false;
00033                 }
00034             }
00035
00036             return true;
00037         }
00038
00039         public object Clone()
00040         {
00041             return new CurrencyWallet { Currencies = Currencies.Select(x =>
    (CurrencyContainer)x.Clone()).ToList() };
00042         }
00043     }
00044 }
```

7.31 CurrencyOperator.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
```

```
00005     public abstract class CurrencyOperator : MonoBehaviour
00006     {
00007         protected CurrencyRegistry MyRegistry;
00008 
00009         void Start()
00010         {
00011             MyRegistry = CurrencyRegistry.Instance;
00012             if (MyRegistry == null)
00013             {
00014                 Debug.LogWarningFormat("{0}: Missing SpriteRenderer for Sprite Response to operate
      on", name);
00015             }
00016         }
00017 
00018         public void Activate()
00019         {
00020             if (MyRegistry == null)
00021                 return;
00022 
00023             AdjustCurrencies();
00024         }
00025 
00026         protected abstract void AdjustCurrencies();
00027     }
00028 }
```

7.32 CurrencyRefillOperator.cs

```
00001 using System.Collections.Generic;
00002 using UnityEngine.Serialization;
00003 
00004 namespace CardHouse
00005 {
00006     public class CurrencyRefillOperator : CurrencyOperator
00007     {
00008         [FormerlySerializedAs("ResourcesToRefill")]
00009         public List<CurrencyScriptable> CurrenciesToRefill;
00010 
00011         protected override void AdjustCurrencies()
00012         {
00013             foreach (var resource in CurrenciesToRefill)
00014             {
00015                 MyRegistry.Refill(resource.name, PhaseManager.Instance.PlayerIndex);
00016             }
00017         }
00018     }
00019 }
```

7.33 IncrementCurrencyOperator.cs

```
00001 using System.Collections.Generic;
00002 using UnityEngine.Serialization;
00003 
00004 namespace CardHouse
00005 {
00006     public class IncrementCurrencyOperator : CurrencyOperator
00007     {
00008         [FormerlySerializedAs("ResourcesToChange")]
00009         public List<CurrencyQuantity> CurrenciesToChange;
00010 
00011         protected override void AdjustCurrencies()
00012         {
00013             foreach (var resource in CurrenciesToChange)
00014             {
00015                 MyRegistry.AdjustCurrency(resource.CurrencyType.Name,
      PhaseManager.Instance.PlayerIndex, resource.Amount);
00016             }
00017         }
00018     }
00019 }
```

7.34 DragAction.cs

```
00001 namespace CardHouse
00002 {
```

```
00003     public enum DragAction
00004     {
00005         None,
00006         Mount,
00007         UseAndDiscard,
00008         UseOnTargetAndDiscard
00009     }
00010 }
```

7.35 DragDetector.cs

```
00001 using UnityEngine.Events;
00002 using UnityEngine.Serialization;
00003
00004 namespace CardHouse
00005 {
00006     public class DragDetector : Toggleable
00007     {
00008         public GateCollection<NoParams> DragGates;
00009         public UnityEvent OnDragStart;
00010
00011         [FormerlySerializedAs("DropGates")]
00012         public GateCollection<DropParams> GroupDropGates;
00013         public GateCollection<TargetCardParams> TargetCardGates;
00014         public UnityEvent OnDragEnd;
00015
00016         void OnMouseDown()
00017         {
00018             if (!IsActive || !DragGates.AllUnlocked(null))
00019                 return;
00020
00021             OnDragStart.Invoke();
00022         }
00023
00024         void OnMouseUp()
00025         {
00026             if (!IsActive || !DragGates.AllUnlocked(null))
00027                 return;
00028
00029             OnDragEnd.Invoke();
00030         }
00031     }
00032 }
```

7.36 Dragging.cs

```
00001 using System;
00002 using UnityEngine;
00003
00004 namespace CardHouse
00005 {
00006     public class Dragging : MonoBehaviour
00007     {
00008         public float DefaultCardZ = 0f;
00009         public float CardPopupDistance = 1f;
00010
00011         public bool UseGrabOffset;
00012         public Vector3 GrabOffset;
00013         public bool SetNewOffsetOnGrab;
00014
00015         public SeekerScriptable<Vector3> DragHomingStrategy;
00016         Seeker<Vector3> MyStrategy;
00017
00018         bool IsDragging;
00019         DragDetector TargetDraggable;
00020         Homing TargetHoming;
00021         Turning TargetTurning;
00022         float StartingZ;
00023
00024         public static Dragging Instance;
00025         public Action<DragDetector> OnDrag;
00026         public Action<DragDetector> OnDrop;
00027         public Action<DragDetector> PostDrop;
00028
00029         private void Awake()
00030         {
00031             Instance = this;
00032             MyStrategy = DragHomingStrategy.GetStrategy();
00033         }
```

```
00034
00035         public void UpdateStrategy()
00036         {
00037             MyStrategy = DragHomingStrategy.GetStrategy();
00038         }
00039
00040         public Homing GetTarget()
00041         {
00042             return IsDragging ? TargetHoming : null;
00043         }
00044
00045         public void BeginDragging(DragDetector draggable, Homing homing, Turning turning, bool
      pointUpWhenDragged = true, float? startingZ = null)
00046         {
00047             TargetDraggable = draggable;
00048             TargetHoming = homing;
00049             TargetTurning = turning;
00050             StartingZ = startingZ ?? DefaultCardZ;
00051             var mouseWorldPosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);
00052             if (SetNewOffsetOnGrab)
00053             {
00054                 GrabOffset = homing.transform.position - mouseWorldPosition;
00055             }
00056             if (pointUpWhenDragged)
00057             {
00058                 TargetTurning.StartSeeking(Camera.main.transform.rotation.eulerAngles.z);
00059             }
00060
00061             IsDragging = true;
00062
00063             OnDrag?.Invoke(draggable);
00064         }
00065
00066         void Update()
00067         {
00068             if (!IsDragging)
00069                 return;
00070
00071             GoToMouse(StartingZ - CardPopupDistance);
00072         }
00073
00074         public void StopDragging()
00075         {
00076             if (!IsDragging)
00077                 return;
00078
00079             IsDragging = false;
00080             GoToMouse(StartingZ);
00081             OnDrop?.Invoke(TargetDraggable);
00082             PostDrop?.Invoke(TargetDraggable);
00083         }
00084
00085         void GoToMouse(float newZ)
00086         {
00087             var mouseWorldPosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);
00088             if (UseGrabOffset)
00089             {
00090                 mouseWorldPosition += GrabOffset;
00091             }
00092             var zCorrection = Vector3.back * (mouseWorldPosition.z - newZ); // apply corrective vector
      to ignore "z" position of mouse
00093             TargetHoming.StartSeeking(mouseWorldPosition + zCorrection, MyStrategy);
00094         }
00095     }
00096 }
```

7.37 DragOperator.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     [RequireComponent(typeof(Homing)), RequireComponent(typeof(Turning)),
      RequireComponent(typeof(Scaling))]
00006     public class DragOperator : MonoBehaviour
00007     {
00008         public DragDetector MyDragDetector;
00009
00010         public DragAction DragAction;
00011         public float DragSwell = 1.2f;
00012         public bool PointUpWhenDragged = true;
00013
00014         public SeekerScriptableSet PresentationSeekers;
```

```
00015
00016         Homing MyHoming;
00017         Turning MyTurning;
00018         Scaling MyScaling;
00019
00020
00021         private void Awake()
00022         {
00023             MyHoming = GetComponent<Homing>();
00024             MyTurning = GetComponent<Turning>();
00025             MyScaling = GetComponent<Scaling>();
00026             if (MyDragDetector == null)
00027             {
00028                 MyDragDetector = GetComponent<DragDetector>();
00029             }
00030         }
00031
00032         public void SetDragState(bool newState)
00033         {
00034             if (MyDragDetector == null)
00035                 return;
00036
00037             if (newState)
00038             {
00039                 if (UseDragSwell)
00040                 {
00041                     MyScaling.StartSeeking(DragSwell);
00042                 }
00043                 Dragging.Instance.BeginDragging(MyDragDetector, MyHoming, MyTurning,
      PointUpWhenDragged);
00044             }
00045             else
00046             {
00047                 if (UseDragSwell)
00048                 {
00049                     MyScaling.StartSeeking(1f);
00050                 }
00051                 Dragging.Instance.StopDragging();
00052             }
00053         }
00054
00055         bool UseDragSwell => DragSwell > 0 && DragSwell != 1;
00056     }
00057 }
```

7.38 EventChain.cs

```
00001 using System.Collections.Generic;
00002 using UnityEngine;
00003 using UnityEngine.Events;
00004
00005 namespace CardHouse
00006 {
00007     public class EventChain : MonoBehaviour
00008     {
00009         public List<TimedEvent> Events = new List<TimedEvent>();
00010
00011         public UnityEvent OnChainFinished;
00012
00013         public void Activate()
00014         {
00015             StartCoroutine(TimedEvent.ExecuteChain(Events, () => OnChainFinished?.Invoke()));
00016         }
00017     }
00018 }
```

7.39 TimedEvent.cs

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using UnityEngine;
00005 using UnityEngine.Events;
00006
00007 namespace CardHouse
00008 {
00009     [System.Serializable]
00010     public class TimedEvent
00011     {
```

```
00012         public float Duration;
00013         public UnityEvent Event;
00014
00015         public IEnumerator ActivateAndDelay()
00016         {
00017             Event.Invoke();
00018             yield return new WaitForSeconds(Duration);
00019         }
00020
00021         public static IEnumerator ExecuteChain(List<TimedEvent> events, Action callback = null)
00022         {
00023             if (events != null)
00024             {
00025                 foreach (var e in events)
00026                 {
00027                     yield return e.ActivateAndDelay();
00028                 }
00029             }
00030
00031             callback?.Invoke();
00032         }
00033     }
00034 }
```

7.40 BlockAllDrops.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     [RequireComponent(typeof(CardGroup))]
00006     public class BlockAllDrops : Gate<DropParams>
00007     {
00008         protected override bool IsUnlockedInternal(DropParams gateParams)
00009         {
00010             return false;
00011         }
00012     }
00013 }
```

7.41 CurrencyGate.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     [RequireComponent(typeof(CurrencyCost))]
00006     public class CurrencyGate : Gate<NoParams>
00007     {
00008         CurrencyCost MyCost;
00009
00010         void Awake()
00011         {
00012             MyCost = GetComponent<CurrencyCost>();
00013         }
00014
00015         protected override bool IsUnlockedInternal(NoParams gateParams)
00016         {
00017             foreach (var resourceCost in MyCost.Cost)
00018             {
00019                 var amountPlayerHas =
      CurrencyRegistry.Instance.GetCurrency(resourceCost.Cost.CurrencyType,
      PhaseManager.Instance.PlayerIndex);
00020                 if (amountPlayerHas < resourceCost.Cost.Amount)
00021                 {
00022                     return false;
00023                 }
00024             }
00025             return true;
00026         }
00027     }
00028 }
```

7.42 Gate.cs

```
00001 namespace CardHouse
```

```
00002 {
00003     public abstract class Gate<T> : Toggleable
00004     {
00005         public bool IsUnlocked(T argObject)
00006         {
00007             if (!IsActive)
00008                 return true;
00009 
00010             return IsUnlockedInternal(argObject);
00011         }
00012 
00013         protected abstract bool IsUnlockedInternal(T argObject);
00014     }
00015 }
```

7.43 GateCollection.cs

```
00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 
00005 namespace CardHouse
00006 {
00007     [Serializable]
00008     public class GateCollection<T>
00009     {
00010         public List<Gate<T» Gates;
00011 
00012         public bool AllUnlocked(T gateParams)
00013         {
00014             return Gates.Count == 0 || Gates.All(x => x.IsUnlocked(gateParams));
00015         }
00016 
00017         public bool AnyUnlocked(T gateParams)
00018         {
00019             return Gates.Count == 0 || Gates.Any(x => x.IsUnlocked(gateParams));
00020         }
00021     }
00022 }
```

7.44 DropParams.cs

```
00001 namespace CardHouse
00002 {
00003     public class DropParams
00004     {
00005         public CardGroup Source;
00006         public CardGroup Target;
00007         public Card Card;
00008         public DragAction DragType;
00009     }
00010 }
```

7.45 NoParams.cs

```
00001 namespace CardHouse
00002 {
00003     public class NoParams
00004     {
00005     }
00006 }
```

7.46 TargetCardParams.cs

```
00001 namespace CardHouse
00002 {
00003     public class TargetCardParams
00004     {
00005         public Card Source;
00006         public Card Target;
00007     }
00008 }
```

7.47 CardGroup.cs

```
00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using UnityEngine;
00005 using UnityEngine.Events;
00006
00007 namespace CardHouse
00008 {
00009     [RequireComponent(typeof(CardGroupSettings))]
00010     public class CardGroup : MonoBehaviour
00011     {
00012         public bool HilightOnCardEntry = true;
00013         public GameObject Hilight;
00014
00015         public GateCollection<DropParams> DropGates;
00016
00017         public SeekerScriptable<Vector3> ShuffleStrategy;
00018
00019         public List<Card> MountedCards = new List<Card>();
00020         CardGroupSettings Strategy;
00021
00022         public UnityEvent OnGroupChanged;
00023
00024         public static Action<CardGroup> OnNewActiveGroup;
00025         public static Action<Card, Card> OnCardUsedOnTarget;
00026
00027         static List<CardGroup> GroupsHoveredWithObjects = new List<CardGroup>();
00028
00029         int CollidersEntered = 0;
00030
00031         public static CardGroup HilightedGroup
00032         {
00033             get
00034             {
00035                 return GroupsHoveredWithObjects.Count > 0 ?
      GroupsHoveredWithObjects[GroupsHoveredWithObjects.Count - 1] : null;
00036             }
00037         }
00038
00039         public static void AddHoveredGroup(CardGroup group)
00040         {
00041             GroupsHoveredWithObjects.Remove(group);
00042             GroupsHoveredWithObjects.Add(group);
00043             OnNewActiveGroup?.Invoke(group);
00044         }
00045
00046         public static void RemoveHoveredGroup(CardGroup group)
00047         {
00048             GroupsHoveredWithObjects.Remove(group);
00049             if (GroupsHoveredWithObjects.Count > 0)
00050             {
00051                 var newActiveGroup = GroupsHoveredWithObjects[GroupsHoveredWithObjects.Count - 1];
00052                 newActiveGroup.SetHilightState(true);
00053                 OnNewActiveGroup?.Invoke(newActiveGroup);
00054             }
00055         }
00056
00057         public static Card GetActiveCard(DragDetector draggable)
00058         {
00059             var closestIndex =
      HilightedGroup.GetClosestMountedCardIndex(draggable.transform.position);
00060             if (closestIndex == null)
00061                 return null;
00062
00063             return HilightedGroup.MountedCards[(int)closestIndex];
00064         }
00065
00066         void Awake()
00067         {
00068             Strategy = GetComponent<CardGroupSettings>();
00069         }
00070
00071         void Start()
00072         {
00073             OnNewActiveGroup += HandleNewActiveGroup;
00074
00075             if (Dragging.Instance == null)
00076             {
00077                 Debug.LogWarning("Groups require the Dragging component on the System prefab, and will
      not function otherwise");
00078                 return;
00079             }
00080
00081             Dragging.Instance.OnDrag += HandleDragStart;
00082             Dragging.Instance.OnDrop += HandleDragDrop;
```

```
00083             Dragging.Instance.PostDrop += HandlePostDrop;
00084         }
00085
00086         void OnDestroy()
00087         {
00088             OnNewActiveGroup -= HandleNewActiveGroup;
00089             if (Dragging.Instance != null)
00090             {
00091                 Dragging.Instance.OnDrag -= HandleDragStart;
00092                 Dragging.Instance.OnDrop -= HandleDragDrop;
00093                 Dragging.Instance.PostDrop -= HandlePostDrop;
00094             }
00095         }
00096
00097         void HandleNewActiveGroup(CardGroup newActiveGroup)
00098         {
00099             if (newActiveGroup != this)
00100             {
00101                 Hilight?.SetActive(false);
00102             }
00103         }
00104
00105         void HandleDragStart(DragDetector draggedCard)
00106         {
00107             CollidersEntered = 0;
00108         }
00109
00110         void HandleDragDrop(DragDetector dragDetector)
00111         {
00112             var cardComponent = dragDetector.GetComponent<Card>();
00113             var cardDragHandler = dragDetector.GetComponent<DragOperator>();
00114             var failedDragFromThisGroup = HilightedGroup == null && cardComponent != null &&
      MountedCards.Contains(cardComponent);
00115             if (failedDragFromThisGroup)
00116             {
00117                 Strategy.Apply(MountedCards);
00118             }
00119
00120             if (HilightedGroup != this)
00121                 return;
00122
00123             if (cardComponent != null && cardDragHandler != null)
00124             {
00125                 var dropParams = new DropParams
00126                 {
00127                     Source = cardComponent?.Group,
00128                     Target = this,
00129                     Card = cardComponent,
00130                     DragType = cardDragHandler == null ? DragAction.None : cardDragHandler.DragAction
00131                 };
00132
00133                 var isTargetable = true;
00134                 if (dropParams.DragType == DragAction.UseOnTargetAndDiscard)
00135                 {
00136                     var closestIndex = GetClosestMountedCardIndex(cardComponent.transform.position);
00137                     if (closestIndex != null)
00138                     {
00139                         var targetCardParams = new TargetCardParams
00140                         {
00141                             Source = cardComponent,
00142                             Target = MountedCards[(int)closestIndex]
00143                         };
00144                         isTargetable =
      targetCardParams.Source.GetComponent<DragDetector>().TargetCardGates.AllUnlocked(targetCardParams)
00145                                        &&
      (targetCardParams.Target.GetComponent<DragDetector>()?.TargetCardGates.AllUnlocked(targetCardParams)
      ?? true);
00146                     }
00147                 }
00148
00149                 if (!DropGates.AllUnlocked(dropParams) ||
      !dragDetector.GroupDropGates.AllUnlocked(dropParams) || !isTargetable) // Return to sender
00150                 {
00151                     cardComponent.Group?.ApplyStrategy();
00152                     return;
00153                 }
00154
00155                 switch (cardComponent.GetComponent<DragOperator>().DragAction)
00156                 {
00157                     case DragAction.Mount:
00158                         int? insertPoint = null;
00159
00160                         switch (Strategy.DragMountingMode)
00161                         {
00162                             case MountingMode.Top:
00163                                 break;
00164                             case MountingMode.Bottom:
```

```
00165                                 insertPoint = 0;
00166                                 break;
00167                             case MountingMode.Closest:
00168                                 var closestIndex =
      GetClosestMountedCardIndex(dragDetector.transform.position);
00169                                 if (closestIndex == null)
00170                                     break;
00171
00172                                 var diff = MountedCards[(int)closestIndex].transform.position -
      dragDetector.transform.position;
00173                                 insertPoint = diff.x > 0 ? closestIndex : closestIndex + 1;
00174                                 break;
00175                         }
00176
00177                         if (cardComponent.GetComponent<CardLoyalty>() != null
00178                             && GroupRegistry.Instance?.GetOwnerIndex(cardComponent.Group) != null
00179                             && GroupRegistry.Instance?.GetOwnerIndex(this) == null)
00180                         {
00181                             cardComponent.GetComponent<CardLoyalty>().PlayerIndex =
      (int)GroupRegistry.Instance.GetOwnerIndex(cardComponent.Group);
00182                         }
00183
00184                         Mount(cardComponent, insertPoint);
00185                         cardComponent.HandlePlayed();
00186                         break;
00187                     case DragAction.UseAndDiscard:
00188                         var discardGroup = GroupRegistry.Instance?.Get(GroupName.Discard,
      PhaseManager.Instance == null ? null : PhaseManager.Instance.PlayerIndex);
00189                         if (discardGroup == null)
00190                         {
00191                             Debug.LogWarningFormat("{0}: Could not find Discard group to discard this
      card", name);
00192                             cardComponent.Group?.ApplyStrategy();
00193                             break;
00194                         }
00195
00196                         var seekerSets = new SeekerSetList();
00197                         var presentationTransform =
      PhaseManager.Instance?.CurrentPhase?.CardPresentationPosition;
00198                         if (presentationTransform != null)
00199                         {
00200                             seekerSets.Add(new SeekerSet
00201                             {
00202                                 Card = cardComponent,
00203                                 Homing =
      cardDragHandler.PresentationSeekers.Homing?.GetStrategy(presentationTransform.position),
00204                                 Turning =
      cardDragHandler.PresentationSeekers.Turning?.GetStrategy(CardHouse.Utils.CorrectAngle(presentationTransform.rotation.eul
00205                                 Scaling =
      cardDragHandler.PresentationSeekers.Scaling?.GetStrategy(presentationTransform.lossyScale.x)
00206                             });
00207                         }
00208                         discardGroup.Mount(cardComponent, seekerSets: seekerSets);
00209                         cardComponent.HandlePlayed();
00210                         break;
00211                     case DragAction.UseOnTargetAndDiscard:
00212                         var discardGroup1 = GroupRegistry.Instance?.Get(GroupName.Discard,
      PhaseManager.Instance == null ? null : PhaseManager.Instance.PlayerIndex);
00213                         var closestIndex1 =
      GetClosestMountedCardIndex(dragDetector.transform.position);
00214                         if (discardGroup1 == null || closestIndex1 == null)
00215                         {
00216                             cardComponent.Group?.ApplyStrategy();
00217                             break;
00218                         }
00219
00220                         var seekerSets1 = new SeekerSetList();
00221                         var presentationTransform1 =
      PhaseManager.Instance?.CurrentPhase?.CardPresentationPosition;
00222                         if (presentationTransform1 != null)
00223                         {
00224                             seekerSets1.Add(new SeekerSet
00225                             {
00226                                 Card = cardComponent,
00227                                 Homing =
      cardDragHandler.PresentationSeekers.Homing?.GetStrategy(presentationTransform1.position),
00228                                 Turning =
      cardDragHandler.PresentationSeekers.Turning?.GetStrategy(CardHouse.Utils.CorrectAngle(presentationTransform1.rotation.eu
00229                                 Scaling =
      cardDragHandler.PresentationSeekers.Scaling?.GetStrategy(presentationTransform1.lossyScale.x)
00230                             });
00231                         }
00232                         discardGroup1.Mount(cardComponent, seekerSets: seekerSets1);
00233                         var targetCard = MountedCards[(int)closestIndex1];
00234                         OnCardUsedOnTarget?.Invoke(cardComponent, targetCard);
00235                         cardComponent.HandlePlayed();
00236                         break;
```

```
00237                 }
00238             }
00239         }
00240
00241         public int? GetClosestMountedCardIndex(Vector3 position)
00242         {
00243             var closestDist = Mathf.Infinity;
00244             int? closestIndex = null;
00245             for (var i = 0; i < MountedCards.Count; i++)
00246             {
00247                 var card = MountedCards[i];
00248                 var diff = card.transform.position - position;
00249                 var dist = diff.magnitude;
00250                 if (dist < closestDist)
00251                 {
00252                     closestDist = dist;
00253                     closestIndex = i;
00254                 }
00255             }
00256             return closestIndex;
00257         }
00258
00259         void HandlePostDrop(DragDetector dragDetector)
00260         {
00261             GroupsHoveredWithObjects.Clear();
00262             Hilight?.SetActive(false);
00263         }
00264
00265         void OnTriggerEnter2D(Collider2D col)
00266         {
00267             HandleTriggerEnter2D(col);
00268         }
00269
00270         public void HandleTriggerEnter2D(Collider2D col)
00271         {
00272             var draggable = col.gameObject.GetComponent<DragOperator>();
00273             if ((draggable == null || draggable.DragAction == DragAction.Mount) && !HasRoom())
00274                 return;
00275
00276             RespondToObjectCrossingBoundary(col, true);
00277         }
00278
00279         void OnTriggerExit2D(Collider2D col)
00280         {
00281             HandleTriggerExit2D(col);
00282         }
00283
00284         public void HandleTriggerExit2D(Collider2D col)
00285         {
00286             RespondToObjectCrossingBoundary(col, false);
00287         }
00288
00289         void RespondToObjectCrossingBoundary(Collider2D col, bool isEntry)
00290         {
00291             var thingBeingDragged = Dragging.Instance?.GetTarget();
00292             var cardComponent = thingBeingDragged?.GetComponent<Card>();
00293             var dragHandler = thingBeingDragged?.GetComponent<DragOperator>();
00294             var dropParams = new DropParams
00295             {
00296                 Source = cardComponent?.Group,
00297                 Target = this,
00298                 Card = cardComponent,
00299                 DragType = dragHandler == null ? DragAction.None : dragHandler.DragAction
00300             };
00301             if (cardComponent != null
00302                 && dragHandler != null
00303                 && DropGates.AllUnlocked(dropParams)
00304                 && dragHandler.GetComponent<DragDetector>().GroupDropGates.AllUnlocked(dropParams))
00305             {
00306                 CollidersEntered += isEntry ? 1 : -1;
00307                 SetAsActiveGroup(CollidersEntered > 0);
00308             }
00309         }
00310
00311         void SetAsActiveGroup(bool newState)
00312         {
00313             SetHilightState(newState);
00314
00315             if (newState)
00316             {
00317                 AddHoveredGroup(this);
00318             }
00319             else
00320             {
00321                 RemoveHoveredGroup(this);
00322             }
00323         }
```

```
00324
00325         public void SetHilightState(bool newState)
00326         {
00327             if (HilightOnCardEntry)
00328             {
00329                 Hilight.SetActive(newState);
00330             }
00331         }
00332
00333         public void ApplyStrategy()
00334         {
00335             Strategy.Apply(MountedCards);
00336         }
00337
00338         public bool HasRoom()
00339         {
00340             return Strategy.CardLimit < 0 || MountedCards.Count < Strategy.CardLimit;
00341         }
00342
00343         public void Mount(Card card, int? index = null, bool instaFlip = false, SeekerSetList
      seekerSets = null, SeekerSet seekersForUnmounting = null)
00344         {
00345             card.Group?.UnMount(card, seekersForUnmounting);
00346
00347             if (index == null || index >= MountedCards.Count)
00348             {
00349                 MountedCards.Add(card);
00350             }
00351             else if (index < 0)
00352             {
00353                 MountedCards.Insert(0, card);
00354             }
00355             else
00356             {
00357                 MountedCards.Insert((int)index, card);
00358             }
00359
00360             card.Group = this;
00361             card.TriggerMountEvents(this);
00362             OnGroupChanged?.Invoke();
00363
00364             Strategy.Apply(MountedCards, instaFlip, seekerSets);
00365         }
00366
00367         public bool SafeMount(Card card, int? index = null)
00368         {
00369             var hasRoom = HasRoom();
00370             if (hasRoom)
00371             {
00372                 Mount(card, index);
00373             }
00374             return hasRoom;
00375         }
00376
00377         public int? UnMount(Card card, SeekerSet seekersForUnmounting = null)
00378         { // returns index of card if found, -1 if not found
00379             int? index = null;
00380             if (MountedCards.Contains(card))
00381             {
00382                 index = MountedCards.IndexOf(card);
00383                 UnMount(index, seekersForUnmounting);
00384             }
00385
00386             return index;
00387         }
00388
00389         public Card UnMount(int? index = null, SeekerSet seekersForUnmounting = null)
00390         {
00391             var card = Get(index);
00392             if (card != null)
00393             {
00394                 MountedCards.Remove(card);
00395                 card.Group = null;
00396                 Strategy.Apply(MountedCards, seekerSets: new SeekerSetList { seekersForUnmounting });
00397
00398                 card.TriggerUnMountEvents(GroupRegistry.Instance?.GetGroupName(this) ??
      GroupName.None);
00399                 OnGroupChanged?.Invoke();
00400             }
00401             return card;
00402         }
00403
00404         public Card Get(int? index = null)
00405         {
00406             if (MountedCards.Count == 0)
00407                 return null;
00408
```

```
00409             Card card;
00410             if (index == null || index >= MountedCards.Count)
00411             {
00412                 card = MountedCards[MountedCards.Count - 1];
00413             }
00414             else if (index < 0)
00415             {
00416                 card = MountedCards[0];
00417             }
00418             else
00419             {
00420                 card = MountedCards[(int)index];
00421             }
00422 
00423             return card;
00424         }
00425 
00426         public List<Card> Get(GroupTargetType targetType, int count)
00427         {
00428             var output = new List<Card>();
00429 
00430             var pool = MountedCards.ToList();
00431             for (int i = 0; i < count; i++)
00432             {
00433                 Card cardToAdd = null;
00434                 if (pool.Count > 0)
00435                 {
00436                     switch (targetType)
00437                     {
00438                         case GroupTargetType.First:
00439                             cardToAdd = pool[0];
00440                             break;
00441                         case GroupTargetType.Last:
00442                             cardToAdd = pool[pool.Count - 1];
00443                             break;
00444                         case GroupTargetType.Random:
00445                             cardToAdd = pool[UnityEngine.Random.Range(0, pool.Count - 1)];
00446                             break;
00447                     }
00448                 }
00449 
00450                 if (cardToAdd != null)
00451                 {
00452                     pool.Remove(cardToAdd);
00453                     output.Add(cardToAdd);
00454                 }
00455 
00456             }
00457             return output;
00458         }
00459 
00460         public int? IndexOf(Card card)
00461         {
00462             if (MountedCards.Contains(card))
00463             {
00464                 return MountedCards.IndexOf(card);
00465             }
00466             return null;
00467         }
00468 
00469         public void Shuffle(bool isInstant = false)
00470         {
00471             var newMountedCards = new List<Card>();
00472             while (MountedCards.Count > 0)
00473             {
00474                 var chosenIndex = UnityEngine.Random.Range(0, MountedCards.Count);
00475                 newMountedCards.Add(MountedCards[chosenIndex]);
00476                 MountedCards.RemoveAt(chosenIndex);
00477             }
00478             MountedCards = newMountedCards;
00479 
00480             var seekerSetList = new SeekerSetList();
00481             if (isInstant)
00482             {
00483                 seekerSetList.Add(new SeekerSet { Homing = new InstantVector3Seeker(), Turning = new
      InstantFloatSeeker(), Scaling = new InstantFloatSeeker() });
00484             }
00485             else
00486             {
00487                 foreach (var card in MountedCards)
00488                 {
00489                     seekerSetList.Add(new SeekerSet { Card = card, Homing =
      ShuffleStrategy?.GetStrategy() ?? new ExponentialVector3Seeker() });
00490                 }
00491             }
00492 
00493             foreach (var card in MountedCards)
```

```
00494             {
00495                 if (card.CanBeUpsideDown)
00496                 {
00497                     card.SetUpsideDown(UnityEngine.Random.Range(0f, 1f) < card.UpsideDownChance);
00498                 }
00499             }
00500 
00501             Strategy.Apply(MountedCards, instaFlip: isInstant, seekerSets: seekerSetList);
00502         }
00503 
00504         public void ShuffleIn(List<Card> cards, bool isInstant = false)
00505         {
00506             if (cards.Count == 0)
00507                 return;
00508 
00509             foreach (var card in cards.ToList())
00510             {
00511                 Mount(card, instaFlip: isInstant);
00512             }
00513             Shuffle(isInstant);
00514         }
00515     }
00516 
00517     public enum GroupTargetType
00518     {
00519         First,
00520         Last,
00521         Random
00522     }
00523 }
```

7.48 GroupInteractability.cs

```
00001 namespace CardHouse
00002 {
00003     public enum GroupInteractability
00004     {
00005         None,
00006         Active,
00007         Inactive,
00008         OnlyTopActive
00009     }
00010 }
```

7.49 GroupName.cs

```
00001 namespace CardHouse
00002 {
00003     public enum GroupName
00004     {
00005         None,
00006         Discard,
00007         Deck,
00008         Hand,
00009         Board,
00010         A,
00011         B,
00012         C,
00013         D
00014     }
00015 }
```

7.50 GroupRegistry.cs

```
00001 using System;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 
00005 namespace CardHouse
00006 {
00007     public class GroupRegistry : MonoBehaviour
00008     {
00009         [Serializable]
00010         public class NamedGroup
00011         {
```

```
00012             public int PlayerIndex;
00013             public GroupName Name;
00014             public CardGroup Group;
00015         }
00016
00017         public List<NamedGroup> Groups = new List<NamedGroup>();
00018
00019         public static GroupRegistry Instance;
00020
00021         void Awake()
00022         {
00023             Instance = this;
00024         }
00025
00026         public CardGroup Get(GroupName name, int? playerIndex)
00027         {
00028             foreach (var group in Groups)
00029             {
00030                 if ((playerIndex == null || group.PlayerIndex == playerIndex) && group.Name == name)
00031                 {
00032                     return group.Group;
00033                 }
00034             }
00035             return null;
00036         }
00037
00038         public GroupName GetGroupName(CardGroup group)
00039         {
00040             foreach (var namedGroup in Groups)
00041             {
00042                 if (namedGroup.Group == group)
00043                 {
00044                     return namedGroup.Name;
00045                 }
00046             }
00047             return GroupName.None;
00048         }
00049
00050         public Loyalty GetLoyalty(CardGroup group, int playerIndex)
00051         {
00052             var ownerIndex = GetOwnerIndex(group);
00053             if (ownerIndex == null)
00054                 return Loyalty.None;
00055
00056             return ownerIndex == playerIndex ? Loyalty.Self : Loyalty.Other;
00057         }
00058
00059         public int? GetOwnerIndex(CardGroup group)
00060         {
00061             foreach (var namedGroup in Groups)
00062             {
00063                 if (namedGroup.Group == group)
00064                 {
00065                     return namedGroup.PlayerIndex;
00066                 }
00067             }
00068
00069             return null;
00070         }
00071     }
00072 }
```

7.51 CardGridLayout.cs

```
00001 using System.Collections.Generic;
00002 using UnityEngine;
00003
00004 namespace CardHouse
00005 {
00006     public class CardGridLayout : CardGroupSettings
00007     {
00008         public int CardsPerRow = 5;
00009         public float MarginalCardOffset = 0.05f;
00010         Collider2D MyCollider;
00011         public bool Straighten = true;
00012
00013         void Awake()
00014         {
00015             MyCollider = GetComponent<Collider2D>();
00016             if (MyCollider == null)
00017             {
00018                 Debug.LogWarningFormat("{0}:{1} needs Collider2D on its game object to function.",
      gameObject.name, "GridLayout");
```

```
00019             }
00020         }
00021
00022         protected override void ApplySpacing(List<Card> cards, SeekerSetList seekerSets)
00023         {
00024             var width = transform.lossyScale.x;
00025             var height = transform.lossyScale.y;
00026
00027             var rowCount = 1 + (cards.Count - 1) / CardsPerRow;
00028             var colSpacing = height / (rowCount + 1);
00029
00030             for (var row = 0; row < rowCount; row++)
00031             {
00032                 var cardsInThisRow = Mathf.Min(CardsPerRow, cards.Count - row * CardsPerRow);
00033                 var rowSpacing = width / (cardsInThisRow + 1);
00034                 for (var col = 0; col < cardsInThisRow; col++)
00035                 {
00036                     var newPos = transform.position
00037                                  + transform.right * width * -0.5f
00038                                  + transform.right * (col + 1) * rowSpacing
00039                                  + transform.up * height * 0.5f
00040                                  + transform.up * (row + 1) * colSpacing * -1
00041                                  + transform.forward * (MountedCardAltitude + MarginalCardOffset *
    (row * CardsPerRow + col)) * -1;
00042
00043                     var cardIndex = row * CardsPerRow + col;
00044                     var card = cards[cardIndex];
00045                     var seekerSet = seekerSets?.GetSeekerSetFor(card);
00046                     card.Homing.StartSeeking(newPos, seekerSet?.Homing);
00047                     if (Straighten)
00048                     {
00049                         card.Turning.StartSeeking(transform.rotation.eulerAngles.z,
    seekerSet?.Turning);
00050                     }
00051                     card.Scaling.StartSeeking(UseMyScale ? transform.lossyScale.y : 1,
    seekerSet?.Scaling);
00052                 }
00053             }
00054         }
00055     }
00056 }
```

7.52 CardGroupSettings.cs

```
00001 using System.Collections.Generic;
00002 using UnityEngine;
00003
00004 namespace CardHouse
00005 {
00006     public abstract class CardGroupSettings : MonoBehaviour
00007     {
00008         public int CardLimit = -1; // Limit < 0 means no limit
00009         public float MountedCardAltitude = 0.01f;
00010         public CardFacing ForcedFacing;
00011         public GroupInteractability ForcedInteractability;
00012         public MountingMode DragMountingMode = MountingMode.Top;
00013         public bool UseMyScale = false;
00014
00015         public void Apply(List<Card> cards, bool instaFlip = false, SeekerSetList seekerSets = null)
00016         {
00017             for (var i = 0; i < cards.Count; i++)
00018             {
00019                 var card = cards[i];
00020                 if (ForcedFacing != CardFacing.None)
00021                 {
00022                     var flipSpeed = 1f;
00023                     if (seekerSets != null && seekerSets.Count > 0 && seekerSets[0] != null)
00024                     {
00025                         flipSpeed = seekerSets[0].FlipSpeed;
00026                     }
00027                     cards[i].SetFacing(ForcedFacing, immediate: instaFlip, spd: flipSpeed);
00028                 }
00029                 if (ForcedInteractability != GroupInteractability.None)
00030                 {
00031                     var col = card.GetComponent<Collider2D>();
00032                     if (col)
00033                     {
00034                         col.enabled = ForcedInteractability == GroupInteractability.Active
00035                                       || ForcedInteractability == GroupInteractability.OnlyTopActive
    && i == cards.Count - 1;
00036                     }
00037                 }
00038             }
```

```
00039
00040             ApplySpacing(cards, seekerSets);
00041         }
00042
00043         protected abstract void ApplySpacing(List<Card> cards, SeekerSetList seekerSets);
00044     }
00045 }
```

7.53 SlotLayout.cs

```
00001 using System.Collections.Generic;
00002 using UnityEngine;
00003
00004 namespace CardHouse
00005 {
00006     public class SlotLayout : CardGroupSettings
00007     {
00008         protected override void ApplySpacing(List<Card> cards, SeekerSetList seekerSets = null)
00009         {
00010             for (var i = 0; i < cards.Count; i++)
00011             {
00012                 var card = cards[i];
00013                 var seekerSet = seekerSets?.GetSeekerSetFor(card);
00014                 card.Homing.StartSeeking(transform.position + Vector3.back * MountedCardAltitude,
      seekerSet?.Homing);
00015                 card.Turning.StartSeeking(transform.rotation.eulerAngles.z, seekerSet?.Turning);
00016                 card.Scaling.StartSeeking(UseMyScale ? transform.lossyScale.y : 1,
      seekerSet?.Scaling);
00017             }
00018         }
00019     }
00020 }
```

7.54 SplayLayout.cs

```
00001 using System.Collections.Generic;
00002 using UnityEngine;
00003
00004 namespace CardHouse
00005 {
00006     public class SplayLayout : CardGroupSettings
00007     {
00008         public float MarginalCardOffset = 0.01f;
00009         public Vector2 ArcCenterOffset = new Vector2(0f, -5f);
00010         [Range(0f, 0.8f)]
00011         public float ArcMargin = 0.3f;
00012         Collider2D MyCollider;
00013
00014         void Awake()
00015         {
00016             MyCollider = GetComponent<Collider2D>();
00017             if (MyCollider == null)
00018             {
00019                 Debug.LogWarningFormat("{0}:{1} needs Collider2D on its game object to function.",
      gameObject.name, "SplayLayout");
00020             }
00021         }
00022
00023         private void Start()
00024         {
00025             ArcCenterOffset = transform.position + transform.right * ArcCenterOffset.x + transform.up
      * ArcCenterOffset.y;
00026         }
00027
00028         protected override void ApplySpacing(List<Card> cards, SeekerSetList seekerSets = null)
00029         {
00030             var width = transform.lossyScale.x * (1f - ArcMargin);
00031             var spacing = width / (cards.Count + 1);
00032             for (var i = 0; i < cards.Count; i++)
00033             {
00034                 var newPos = transform.position
00035                              + Vector3.back * (MountedCardAltitude + i * MarginalCardOffset)
00036                              + transform.right * width * -0.5f
00037                              + transform.right * (i + 1) * spacing;
00038
00039                 var seekerSet = seekerSets?.GetSeekerSetFor(cards[i]);
00040                 cards[i].Homing.StartSeeking(newPos, seekerSet?.Homing);
00041
```

```
00042                 var newAngle = Mathf.Atan2(newPos.y - ArcCenterOffset.y, newPos.x - ArcCenterOffset.x)
      * Mathf.Rad2Deg - 90;
00043                 cards[i].Turning.StartSeeking(newAngle, seekerSet?.Turning);
00044
00045                 cards[i].Scaling.StartSeeking(UseMyScale ? transform.lossyScale.y : 1,
      seekerSet?.Scaling);
00046             }
00047         }
00048     }
00049 }
```

7.55 StackLayout.cs

```
00001 using System.Collections.Generic;
00002 using UnityEngine;
00003
00004 namespace CardHouse
00005 {
00006     public class StackLayout : CardGroupSettings
00007     {
00008         public Vector3 MarginalCardOffset = new Vector3(0.01f, 0.01f, -0.01f);
00009         public TriggerEnterRelay SecondaryCollider;
00010         public bool Straighten = true;
00011
00012         protected override void ApplySpacing(List<Card> cards, SeekerSetList seekerSets = null)
00013         {
00014             for (var i = 0; i < cards.Count; i++)
00015             {
00016                 var seekerSet = seekerSets?.GetSeekerSetFor(cards[i]);
00017                 cards[i].Homing.StartSeeking(transform.position + Vector3.back * MountedCardAltitude +
      MarginalCardOffset * i, seekerSet?.Homing);
00018                 if (Straighten)
00019                 {
00020                     cards[i].Turning.StartSeeking(transform.rotation.eulerAngles.z,
      seekerSet?.Turning);
00021                 }
00022                 cards[i].Scaling.StartSeeking(UseMyScale ? transform.lossyScale.y : 1,
      seekerSet?.Scaling);
00023             }
00024
00025             if (SecondaryCollider != null && cards.Count > 0)
00026             {
00027                 SecondaryCollider.transform.position = transform.position + Vector3.forward +
      MarginalCardOffset * (cards.Count - 1);
00028             }
00029         }
00030     }
00031 }
```

7.56 MountDetector.cs

```
00001 using UnityEngine;
00002 using UnityEngine.Events;
00003
00004 namespace CardHouse
00005 {
00006     [RequireComponent(typeof(Card))]
00007     public class MountDetector : MonoBehaviour
00008     {
00009         public UnityEvent OnMount;
00010         Card MyCard;
00011
00012         void Start()
00013         {
00014             MyCard = GetComponent<Card>();
00015             MyCard.OnMount += HandleMount;
00016         }
00017
00018         void OnDestroy()
00019         {
00020             MyCard.OnMount -= HandleMount;
00021         }
00022
00023         void HandleMount(Card card, CardGroup group)
00024         {
00025             OnMount?.Invoke();
00026         }
00027     }
00028 }
```

7.57 MountingMode.cs

```
00001 namespace CardHouse
00002 {
00003     public enum MountingMode
00004     {
00005         Top,
00006         Bottom,
00007         Closest
00008     }
00009 }
```

7.58 DragTransition.cs

```
00001 using System;
00002
00003 namespace CardHouse
00004 {
00005     [Serializable]
00006     public class DragTransition : GroupTransition
00007     {
00008         public DragAction DragAction;
00009     }
00010 }
```

7.59 GroupTransition.cs

```
00001 using System;
00002
00003 namespace CardHouse
00004 {
00005     [Serializable]
00006     public class GroupTransition
00007     {
00008         public CardGroup Source;
00009         public CardGroup Destination;
00010     }
00011 }
```

7.60 HoverDetector.cs

```
00001 using UnityEngine.Events;
00002
00003 namespace CardHouse
00004 {
00005     public class HoverDetector : Toggleable
00006     {
00007         public UnityEvent OnHover;
00008         public UnityEvent OnUnHover;
00009
00010         void OnMouseEnter()
00011         {
00012             if (!IsActive)
00013                 return;
00014
00015             OnHover.Invoke();
00016         }
00017
00018         void OnMouseExit()
00019         {
00020             if (!IsActive)
00021                 return;
00022
00023             OnUnHover.Invoke();
00024         }
00025     }
00026 }
```

7.61 TriggerEnterRelay.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     public class TriggerEnterRelay : Toggleable
00006     {
00007         public CardGroup Relay;
00008
00009         void OnTriggerEnter2D(Collider2D col)
00010         {
00011             if (!IsActive)
00012                 return;
00013
00014             Relay.HandleTriggerEnter2D(col);
00015         }
00016
00017         void OnTriggerExit2D(Collider2D col)
00018         {
00019             if (!IsActive)
00020                 return;
00021
00022             Relay.HandleTriggerExit2D(col);
00023         }
00024     }
00025 }
```

7.62 LifetimeDestructor.cs

```
00001 using System.Collections;
00002 using UnityEngine;
00003
00004 namespace CardHouse
00005 {
00006     public class LifetimeDestructor : MonoBehaviour
00007     {
00008         public float Lifetime = 1f;
00009
00010         void Start()
00011         {
00012             StartCoroutine(DieAfterTime(Lifetime));
00013         }
00014
00015         IEnumerator DieAfterTime(float delay)
00016         {
00017             yield return new WaitForSeconds(delay);
00018             Destroy(gameObject);
00019         }
00020     }
00021 }
```

7.63 CardDropGateDimmer.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     [RequireComponent(typeof(Card), typeof(DragDetector))]
00006     public class CardDropGateDimmer : Toggleable
00007     {
00008         public MultiSpriteOperator Handler;
00009         public string ActiveMessage;
00010         public string InactiveMessage;
00011
00012         DragDetector ThingBeingDragged;
00013         bool IsGroupTargetable;
00014         bool AmITargetable;
00015
00016         Dragging MyDragging;
00017         Card MyCard;
00018         DragDetector MyDragDetector;
00019
00020         void Start()
00021         {
00022             MyCard = GetComponent<Card>();
00023             MyDragDetector = GetComponent<DragDetector>();
00024             MyDragging = Dragging.Instance;
00025             if (MyDragging == null)
```

```
00026             {
00027                 Debug.LogError("Drag Target Drop Dimmers require a Dragging to operate!");
00028                 return;
00029             }
00030 
00031             MyDragging.OnDrag += HandleDrag;
00032             MyDragging.OnDrop += HandleDrop;
00033         }
00034 
00035         void OnDestroy()
00036         {
00037             if (MyDragging != null)
00038             {
00039                 MyDragging.OnDrag -= HandleDrag;
00040                 MyDragging.OnDrop -= HandleDrop;
00041             }
00042         }
00043 
00044         void HandleDrag(DragDetector draggable)
00045         {
00046             if (!IsActive || Handler == null)
00047                 return;
00048 
00049             var draggedCard = draggable.GetComponent<Card>();
00050             var dragHandler = draggable.GetComponent<DragOperator>();
00051             if (draggedCard == null || dragHandler == null || dragHandler.DragAction !=
      DragAction.UseOnTargetAndDiscard)
00052                 return;
00053 
00054             ThingBeingDragged = draggable;
00055 
00056             var dropParams = new DropParams
00057             {
00058                 Source = draggedCard?.Group,
00059                 Target = MyCard.Group,
00060                 Card = draggedCard,
00061                 DragType = dragHandler == null ? DragAction.None : dragHandler.DragAction
00062             };
00063             IsGroupTargetable = draggable.GroupDropGates.AllUnlocked(dropParams)
00064                                 && MyCard.Group.DropGates.AllUnlocked(dropParams);
00065 
00066             AmITargetable = true;
00067             if (dropParams.DragType == DragAction.UseOnTargetAndDiscard)
00068             {
00069                 var targetCardParams = new TargetCardParams
00070                 {
00071                     Source = draggedCard,
00072                     Target = MyCard
00073                 };
00074                 AmITargetable =
      targetCardParams.Source.GetComponent<DragDetector>().TargetCardGates.AllUnlocked(targetCardParams)
00075                                 &&
      targetCardParams.Target.GetComponent<DragDetector>().TargetCardGates.AllUnlocked(targetCardParams);
00076             }
00077 
00078             Handler.Activate(IsGroupTargetable && AmITargetable ? ActiveMessage : InactiveMessage,
      this);
00079         }
00080 
00081         void Update()
00082         {
00083             if (ThingBeingDragged == null)
00084                 return;
00085 
00086             if (MyCard.Group == CardGroup.HilightedGroup)
00087             {
00088                 Handler.Activate((AmITargetable && CardGroup.GetActiveCard(ThingBeingDragged) ==
      MyCard) ? ActiveMessage : InactiveMessage, this);
00089             }
00090             else if (ThingBeingDragged != MyDragDetector)
00091             {
00092                 Handler.Activate(IsGroupTargetable && AmITargetable ? ActiveMessage : InactiveMessage,
      this);
00093             }
00094         }
00095 
00096         void HandleDrop(DragDetector draggable)
00097         {
00098             if (!IsActive || Handler == null)
00099                 return;
00100 
00101             ThingBeingDragged = null;
00102             Handler.Activate(ActiveMessage, this);
00103         }
00104     }
00105 }
```

7.64 CardLoyalty.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     public class CardLoyalty : MonoBehaviour
00006     {
00007         public int PlayerIndex;
00008     }
00009 }
```

7.65 DragGateDimmer.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     [RequireComponent(typeof(DragDetector))]
00006     public class DragGateDimmer : Toggleable
00007     {
00008         public MultiSpriteOperator Handler;
00009         public string ActiveMessage;
00010         public string InactiveMessage;
00011
00012         DragDetector MyDraggable;
00013
00014         void Start()
00015         {
00016             MyDraggable = GetComponent<DragDetector>();
00017         }
00018
00019         public void UpdateHandler()
00020         {
00021             if (!IsActive || Handler == null)
00022                 return;
00023
00024             Handler.Activate(
00025                 MyDraggable.DragGates.AllUnlocked(null)
00026                     ? ActiveMessage
00027                     : InactiveMessage,
00028                 this);
00029         }
00030     }
00031 }
```

7.66 GroupDropGateDimmer.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     [RequireComponent(typeof(CardGroup))]
00006     public class GroupDropGateDimmer : MonoBehaviour
00007     {
00008         public SpriteOperator Handler;
00009         public string ActiveMessage;
00010         public string InactiveMessage;
00011
00012         CardGroup MyGroup;
00013         Dragging MyDragging;
00014
00015         void Start()
00016         {
00017             MyGroup = GetComponent<CardGroup>();
00018             MyDragging = Dragging.Instance;
00019             if (MyDragging == null)
00020             {
00021                 Debug.LogError("Droppability Dimmers need the Dragging script to exist!");
00022                 return;
00023             }
00024
00025             MyDragging.OnDrag += HandleDrag;
00026             MyDragging.OnDrop += HandleDrop;
00027         }
00028
00029         void OnDestroy()
00030         {
00031             if (MyDragging != null)
```

```
00032             {
00033                 MyDragging.OnDrag -= HandleDrag;
00034                 MyDragging.OnDrop -= HandleDrop;
00035             }
00036         }
00037
00038         void HandleDrag(DragDetector dragDetector)
00039         {
00040             if (Handler == null)
00041                 return;
00042
00043             var draggable = dragDetector.GetComponent<DragOperator>();
00044             var card = dragDetector.GetComponent<Card>();
00045             if (draggable == null || card == null)
00046             {
00047                 Debug.LogWarningFormat("{0}: Dropped object {1} needs DragHandler and Card components
      to use PhaseDroppabilityDimmer", gameObject, dragDetector.gameObject);
00048                 return;
00049             }
00050
00051             var dropParams = new DropParams
00052             {
00053                 Source = card?.Group,
00054                 Target = MyGroup,
00055                 Card = card,
00056                 DragType = draggable == null ? DragAction.None : draggable.DragAction
00057             };
00058
00059             var gatesUnlocked = MyGroup.DropGates.AllUnlocked(dropParams)
00060                                 && dragDetector.GroupDropGates.AllUnlocked(dropParams);
00061
00062             Handler.Activate(
00063                 gatesUnlocked && MyGroup.HasRoom()
00064                     ? ActiveMessage
00065                     : InactiveMessage,
00066                 this);
00067         }
00068
00069         void HandleDrop(DragDetector draggable)
00070         {
00071             if (Handler == null)
00072                 return;
00073
00074             Handler.Activate(InactiveMessage, this);
00075         }
00076     }
00077 }
```

7.67 Loyalty.cs

```
00001 using System;
00002
00003 namespace CardHouse
00004 {
00005     [Flags]
00006     public enum Loyalty
00007     {
00008         None = 0,
00009         Self = 1,
00010         Other = 2
00011     }
00012 }
```

7.68 LoyaltyGateCardDrop.cs

```
00001 using System.Collections.Generic;
00002 using UnityEngine;
00003
00004 namespace CardHouse
00005 {
00006     [RequireComponent(typeof(Card)), RequireComponent(typeof(DragDetector))]
00007     public class LoyaltyGateCardDrop : Gate<DropParams>
00008     {
00009         public Loyalty Loyalty;
00010         public List<GroupName> Destinations;
00011
00012         void Awake()
00013         {
00014         }
```

```
00015
00016         protected override bool IsUnlockedInternal(DropParams gateParams)
00017         {
00018             var groupLoyalty = GroupRegistry.Instance.GetLoyalty(gateParams.Target,
    PhaseManager.Instance.PlayerIndex);
00019             return (Loyalty & groupLoyalty) != 0;
00020         }
00021     }
00022 }
```

7.69 Phase.cs

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using System.Linq;
00005 using UnityEngine;
00006 using UnityEngine.UI;
00007
00008 namespace CardHouse
00009 {
00010     [Serializable]
00011     public class Phase
00012     {
00013         public string Name;
00014         public int PlayerIndex;
00015         public Transform CameraPosition;
00016         public Transform CardPresentationPosition;
00017         public List<Button> ActiveButtons;
00018         public List<ClickDetector> ValidClickTargets;
00019         public List<DragTransition> ValidDrags;
00020         public List<TimedEvent> OnPhaseStartEventChain;
00021         public List<TimedEvent> OnPhaseEndEventChain;
00022
00023         public IEnumerator Start()
00024         {
00025             PhaseManager.Instance?.SetCameraPosition(CameraPosition);
00026             yield return TimedEvent.ExecuteChain(OnPhaseStartEventChain);
00027             foreach (var button in ActiveButtons)
00028             {
00029                 button.interactable = true;
00030             }
00031         }
00032
00033         public IEnumerator End()
00034         {
00035             yield return TimedEvent.ExecuteChain(OnPhaseEndEventChain);
00036         }
00037
00038         public bool IsValidDragStart(CardGroup source, DragAction dragAction)
00039         {
00040             return ValidDrags.Any(x => x.Source == source && x.DragAction == dragAction);
00041         }
00042
00043         public bool IsValidDrag(CardGroup source, CardGroup destination, DragAction dragAction)
00044         {
00045             return ValidDrags.Any(x => x.Source == source && x.Destination == destination &&
    x.DragAction == dragAction);
00046         }
00047     }
00048 }
```

7.70 PhaseChangeDetector.cs

```
00001 using UnityEngine;
00002 using UnityEngine.Events;
00003
00004 namespace CardHouse
00005 {
00006     public class PhaseChangeDetector : MonoBehaviour
00007     {
00008         public UnityEvent OnPhaseChange;
00009
00010         PhaseManager MyPhaseManager;
00011
00012         void Start()
00013         {
00014             MyPhaseManager = PhaseManager.Instance;
00015             if (MyPhaseManager != null)
```

```
00016             {
00017                 MyPhaseManager.OnPhaseChanged += HandlePhaseChanged;
00018             }
00019         }
00020
00021         void OnDestroy()
00022         {
00023             if (MyPhaseManager != null)
00024             {
00025                 MyPhaseManager.OnPhaseChanged -= HandlePhaseChanged;
00026             }
00027         }
00028
00029         void HandlePhaseChanged(Phase phase)
00030         {
00031             OnPhaseChange.Invoke();
00032         }
00033     }
00034 }
```

7.71 PhaseGateCardDragStart.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     [RequireComponent(typeof(Card)), RequireComponent(typeof(DragOperator))]
00006     public class PhaseGateCardDragStart : Gate<NoParams>
00007     {
00008         Card MyCard;
00009         DragOperator MyDraggable;
00010
00011         void Awake()
00012         {
00013             MyCard = GetComponent<Card>();
00014             MyDraggable = GetComponent<DragOperator>();
00015         }
00016
00017         protected override bool IsUnlockedInternal(NoParams gateParams)
00018         {
00019             return PhaseManager.Instance.IsValidDragStart(MyCard.Group, MyDraggable.DragAction);
00020         }
00021     }
00022 }
```

7.72 PhaseGateCardDrop.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     [RequireComponent(typeof(CardGroup))]
00006     public class PhaseGateCardDrop : Gate<DropParams>
00007     {
00008         CardGroup MyGroup;
00009
00010         void Awake()
00011         {
00012             MyGroup = GetComponent<CardGroup>();
00013         }
00014
00015         protected override bool IsUnlockedInternal(DropParams gateParams)
00016         {
00017             return PhaseManager.Instance?.IsValidDrag(gateParams.Source, MyGroup, gateParams.DragType)
    ?? true;
00018         }
00019     }
00020 }
```

7.73 PhaseManager.cs

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

```
00005 using UnityEngine.UI;
00006
00007 namespace CardHouse
00008 {
00009     public class PhaseManager : MonoBehaviour
00010     {
00011         public List<Button> AllPhaseDependentButtons;
00012         public List<Phase> Phases;
00013         public Phase CurrentPhase => (CurrentPhaseIndex >= 0 && CurrentPhaseIndex < Phases.Count) ?
      Phases[CurrentPhaseIndex] : null;
00014         int CurrentPhaseIndex = 0;
00015         public int PlayerIndex
00016         {
00017             get { return CurrentPhase.PlayerIndex; }
00018         }
00019
00020         public Action<Phase> OnPhaseChanged;
00021         public static PhaseManager Instance;
00022
00023         void Awake()
00024         {
00025             Instance = this;
00026         }
00027
00028         IEnumerator Start()
00029         {
00030             yield return new WaitForEndOfFrame();
00031             if (CurrentPhase != null)
00032             {
00033                 StartCoroutine(CurrentPhase.Start());
00034             }
00035         }
00036
00037         public void HardReset()
00038         {
00039             StartCoroutine(HardResetRoutine());
00040         }
00041
00042         IEnumerator HardResetRoutine()
00043         {
00044             CurrentPhaseIndex = 0;
00045             yield return new WaitForEndOfFrame();
00046             if (CurrentPhase != null)
00047             {
00048                 StartCoroutine(CurrentPhase.Start());
00049                 OnPhaseChanged?.Invoke(CurrentPhase);
00050             }
00051         }
00052
00053         public void NextPhase()
00054         {
00055             foreach (var button in AllPhaseDependentButtons)
00056             {
00057                 button.interactable = false;
00058             }
00059             StartCoroutine(PhaseTransition());
00060         }
00061
00062         IEnumerator PhaseTransition()
00063         {
00064             yield return CurrentPhase.End();
00065             CurrentPhaseIndex = (CurrentPhaseIndex + 1) % Phases.Count;
00066             yield return CurrentPhase.Start();
00067             OnPhaseChanged?.Invoke(CurrentPhase);
00068         }
00069
00070         public bool IsValidDragStart(CardGroup source, DragAction dragAction)
00071         {
00072             if (CurrentPhase == null)
00073                 return true;
00074
00075             return CurrentPhase.IsValidDragStart(source, dragAction);
00076         }
00077
00078         public bool IsValidDrag(CardGroup source, CardGroup destination, DragAction dragAction)
00079         {
00080             if (CurrentPhase == null)
00081                 return true;
00082
00083             return CurrentPhase.IsValidDrag(source, destination, dragAction);
00084         }
00085
00086         public bool IsValidClick(ClickDetector blutton)
00087         {
00088             if (CurrentPhase == null)
00089                 return true;
00090
```

```
00091             return CurrentPhase.ValidClickTargets.Contains(blutton);
00092         }
00093
00094         public void SetCameraPosition(Transform cameraPosition)
00095         {
00096             if (cameraPosition != null)
00097             {
00098                 var homing = Camera.main.GetComponent<Homing>();
00099                 var turning = Camera.main.GetComponent<Turning>();
00100
00101                 if (homing != null && turning != null)
00102                 {
00103                     homing.StartSeeking(cameraPosition.position + Vector3.forward * (-10 -
      cameraPosition.position.z));
00104                     turning.StartSeeking(cameraPosition.rotation.eulerAngles.z);
00105                 }
00106             }
00107         }
00108
00109         void Update()
00110         {
00111 #if DEBUG
00112             if (CurrentPhase?.ValidDrags == null)
00113                 return;
00114
00115             foreach (var validDrag in CurrentPhase.ValidDrags)
00116             {
00117                 var offset = Vector3.zero;
00118                 var color = Color.green;
00119                 switch (validDrag.DragAction)
00120                 {
00121                     case DragAction.UseAndDiscard:
00122                         offset += Vector3.one * 0.1f;
00123                         color = Color.white;
00124                         break;
00125                     case DragAction.UseOnTargetAndDiscard:
00126                         offset += Vector3.one * 0.2f;
00127                         color = Color.cyan;
00128                         break;
00129                 }
00130                 Debug.DrawLine(
00131                     validDrag.Source.transform.position + offset,
00132                     validDrag.Destination.transform.position + offset,
00133                     color);
00134             }
00135 #endif
00136         }
00137     }
00138 }
```

7.74 BaseSeekerComponent.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     public abstract class BaseSeekerComponent<T> : MonoBehaviour
00006     {
00007         protected Seeker<T> MyStrategy;
00008
00009         protected bool IsSeeking;
00010         protected bool UseLocalSpace;
00011
00012         public SeekerScriptable<T> Strategy;
00013
00014         void Awake()
00015         {
00016             MyStrategy = Strategy?.GetStrategy() ?? GetDefaultSeeker();
00017         }
00018
00019         public void StartSeeking(T destination, Seeker<T> strategy = null, bool useLocalSpace = false)
00020         {
00021             IsSeeking = true;
00022             UseLocalSpace = useLocalSpace;
00023             MyStrategy = strategy?.MakeCopy() ?? Strategy?.GetStrategy() ?? GetDefaultSeeker();
00024             MyStrategy.StartSeeking(GetCurrentValue(), destination);
00025         }
00026
00027         void Update()
00028         {
00029             if (!IsSeeking)
00030                 return;
00031
```

```
00032             var newValue = MyStrategy.Pump(GetCurrentValue(), Time.deltaTime);
00033             SetNewValue(newValue);
00034 
00035             if (MyStrategy.IsDone(newValue))
00036             {
00037                 SetNewValue(MyStrategy.End);
00038                 IsSeeking = false;
00039             }
00040         }
00041 
00042         protected abstract Seeker<T> GetDefaultSeeker();
00043 
00044         protected abstract T GetCurrentValue();
00045 
00046         protected abstract void SetNewValue(T value);
00047     }
00048 }
```

7.75 Homing.cs

```
00001 using UnityEngine;
00002 
00003 namespace CardHouse
00004 {
00005     public class Homing : BaseSeekerComponent<Vector3>
00006     {
00007         protected override Seeker<Vector3> GetDefaultSeeker()
00008         {
00009             return new ExponentialVector3Seeker();
00010         }
00011 
00012         protected override Vector3 GetCurrentValue()
00013         {
00014             return UseLocalSpace ? transform.localPosition : transform.position;
00015         }
00016 
00017         protected override void SetNewValue(Vector3 value)
00018         {
00019             if (UseLocalSpace)
00020             {
00021                 transform.localPosition = value;
00022             }
00023             else
00024             {
00025                 transform.position = value;
00026             }
00027         }
00028     }
00029 }
```

7.76 Scaling.cs

```
00001 
00002 using UnityEngine;
00003 
00004 namespace CardHouse
00005 {
00006     public class Scaling : BaseSeekerComponent<float>
00007     {
00008         protected override Seeker<float> GetDefaultSeeker()
00009         {
00010             return new ExponentialAngleFloatSeeker();
00011         }
00012 
00013         protected override float GetCurrentValue()
00014         {
00015             return UseLocalSpace ? transform.localScale.x : transform.lossyScale.x;
00016         }
00017 
00018         protected override void SetNewValue(float value)
00019         {
00020             if (!UseLocalSpace && transform.parent != null)
00021             {
00022                 transform.localScale = Vector3.one * value / transform.parent.lossyScale.x;
00023             }
00024             else
00025             {
00026                 transform.localScale = Vector3.one * value;
00027             }
```

```
00028         }
00029     }
00030 }
```

7.77 Turning.cs

```
00001
00002 using UnityEngine;
00003
00004 namespace CardHouse
00005 {
00006     public class Turning : BaseSeekerComponent<float>
00007     {
00008         protected override Seeker<float> GetDefaultSeeker()
00009         {
00010             return new ExponentialAngleFloatSeeker();
00011         }
00012
00013         protected override float GetCurrentValue()
00014         {
00015             return UseLocalSpace ? transform.localRotation.eulerAngles.z :
      transform.rotation.eulerAngles.z;
00016         }
00017
00018         protected override void SetNewValue(float value)
00019         {
00020             if (UseLocalSpace)
00021             {
00022                 transform.localRotation = Quaternion.Euler(0, 0, value);
00023             }
00024             else
00025             {
00026                 transform.rotation = Quaternion.Euler(0, 0, value);
00027             }
00028         }
00029     }
00030 }
```

7.78 AnimCurveFloatSeeker.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     public class AnimCurveFloatSeeker : Seeker<float>
00006     {
00007         public float Duration;
00008         protected float Timer;
00009         public AnimationCurve ProgressCurve;
00010
00011         public AnimCurveFloatSeeker(float duration, AnimationCurve progressCurve)
00012         {
00013             Duration = duration;
00014             ProgressCurve = progressCurve;
00015             Timer = 0f;
00016         }
00017
00018         public override Seeker<float> MakeCopy()
00019         {
00020             return new AnimCurveFloatSeeker(Duration, ProgressCurve);
00021         }
00022
00023         public override float Pump(float currentValue, float TimeSinceLastFrame)
00024         {
00025             Timer += TimeSinceLastFrame;
00026             var normalizedTime = Timer / Duration;
00027             return Start + (End - Start) * ProgressCurve.Evaluate(normalizedTime);
00028         }
00029
00030         public override bool IsDone(float currentValue)
00031         {
00032             return Timer >= Duration;
00033         }
00034
00035
00036     }
00037 }
```

7.79 AnimCurveFloatSeekerScriptable.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     [CreateAssetMenu(menuName = "CardHouse/Seekers/Float/Animated Curve")]
00006     public class AnimCurveFloatSeekerScriptable : SeekerScriptable<float>
00007     {
00008         public float Duration = 2f;
00009         public AnimationCurve ProgressCurve;
00010
00011         public override Seeker<float> GetStrategy(params object[] args)
00012         {
00013             return new AnimCurveFloatSeeker(Duration, ProgressCurve);
00014         }
00015     }
00016 }
```

7.80 ExponentialAngleFloatSeeker.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     public class ExponentialAngleFloatSeeker : ExponentialFloatSeeker
00006     {
00007         public ExponentialAngleFloatSeeker(float gain = 8f, float arrivalDist = 0.01f) : base(gain,
      arrivalDist)
00008         {
00009         }
00010
00011         public override float Pump(float currentValue, float TimeSinceLastFrame)
00012         {
00013             return Mathf.LerpAngle(currentValue, End, Gain * TimeSinceLastFrame);
00014         }
00015     }
00016 }
```

7.81 ExponentialAngleFloatSeekerScriptable.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     [CreateAssetMenu(menuName = "CardHouse/Seekers/Float/Exponential (angle)")]
00006     public class ExponentialAngleFloatSeekerScriptable : SeekerScriptable<float>
00007     {
00008         public float Gain = 8f;
00009         public float ArrivalDistance = 0.01f;
00010
00011         public override Seeker<float> GetStrategy(params object[] args)
00012         {
00013             return new ExponentialAngleFloatSeeker(Gain, ArrivalDistance);
00014         }
00015     }
00016 }
```

7.82 ExponentialFloatSeeker.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     public class ExponentialFloatSeeker : Seeker<float>
00006     {
00007         protected float Gain;
00008         protected float ArrivalDistance;
00009
00010         public ExponentialFloatSeeker(float gain = 8f, float arrivalDist = 0.01f)
00011         {
00012             Gain = gain;
00013             ArrivalDistance = arrivalDist;
00014         }
00015
```

```
00016         public override Seeker<float> MakeCopy()
00017         {
00018             return new ExponentialFloatSeeker(Gain, ArrivalDistance);
00019         }
00020 
00021         public override float Pump(float currentValue, float TimeSinceLastFrame)
00022         {
00023             return Mathf.Lerp(currentValue, End, Gain * TimeSinceLastFrame);
00024         }
00025 
00026         public override bool IsDone(float currentValue)
00027         {
00028             return Mathf.Abs(currentValue - End) <= ArrivalDistance;
00029         }
00030     }
00031 }
```

7.83 ExponentialFloatSeekerScriptable.cs

```
00001 using UnityEngine;
00002 
00003 namespace CardHouse
00004 {
00005     [CreateAssetMenu(menuName = "CardHouse/Seekers/Float/Exponential")]
00006     public class ExponentialFloatSeekerScriptable : SeekerScriptable<float>
00007     {
00008         public float Gain = 8f;
00009         public float ArrivalDistance = 0.01f;
00010 
00011         public override Seeker<float> GetStrategy(params object[] args)
00012         {
00013             return new ExponentialFloatSeeker(Gain, ArrivalDistance);
00014         }
00015     }
00016 }
```

7.84 InstantFloatSeeker.cs

```
00001 namespace CardHouse
00002 {
00003     public class InstantFloatSeeker : Seeker<float>
00004     {
00005         public override Seeker<float> MakeCopy()
00006         {
00007             return new InstantFloatSeeker();
00008         }
00009 
00010         public override float Pump(float currentValue, float TimeSinceLastFrame)
00011         {
00012             return End;
00013         }
00014 
00015         public override bool IsDone(float currentValue)
00016         {
00017             return true;
00018         }
00019     }
00020 }
```

7.85 WaypointCurveFloatAngleSeeker.cs

```
00001 using System.Collections.Generic;
00002 using System.Linq;
00003 using UnityEngine;
00004 
00005 namespace CardHouse
00006 {
00007     public class WaypointCurveFloatAngleSeeker : AnimCurveFloatSeeker
00008     {
00009         List<float> Waypoints;
00010 
00011         public WaypointCurveFloatAngleSeeker(float duration, AnimationCurve progressCurve, List<float>
      waypoints) : base(duration, progressCurve)
00012         {
00013             Start = CardHouse.Utils.CorrectAngle(Start);
```

```
00014             End = CardHouse.Utils.CorrectAngle(End);
00015             Waypoints = waypoints.ToList();
00016             for (var i = 0; i < waypoints.Count; i++)
00017             {
00018                 waypoints[i] = CardHouse.Utils.CorrectAngle(waypoints[i]);
00019             }
00020         }
00021 
00022         public override Seeker<float> MakeCopy()
00023         {
00024             return new WaypointCurveFloatSeeker(Duration, ProgressCurve, Waypoints);
00025         }
00026 
00027         public override float Pump(float currentValue, float TimeSinceLastFrame)
00028         {
00029             Timer += TimeSinceLastFrame;
00030             var normalizedTime = Timer / Duration;
00031             var progress = Mathf.Min(ProgressCurve.Evaluate(normalizedTime), 0.9999f);
00032             var destIndex = Mathf.FloorToInt(progress * (Waypoints.Count + 1));
00033             var dest = destIndex >= Waypoints.Count ? End : Waypoints[destIndex];
00034             var lastWaypoint = destIndex == 0 ? Start : Waypoints[destIndex - 1];
00035             return Mathf.LerpAngle(lastWaypoint, dest, (progress % (1f / (Waypoints.Count + 1))) *
    (Waypoints.Count + 1));
00036         }
00037 
00038         public override bool IsDone(float currentValue)
00039         {
00040             return Timer >= Duration;
00041         }
00042     }
00043 }
```

7.86 WaypointCurveFloatAngleSeekerScriptable.cs

```
00001 using System.Collections.Generic;
00002 using UnityEngine;
00003 
00004 namespace CardHouse
00005 {
00006     [CreateAssetMenu(menuName = "CardHouse/Seekers/Float/Waypoint Curve (angle)")]
00007     public class WaypointCurveFloatAngleSeekerScriptable : SeekerScriptable<float>
00008     {
00009         public float Duration = 2f;
00010         public AnimationCurve ProgressCurve;
00011 
00012         public override Seeker<float> GetStrategy(params object[] args)
00013         {
00014             var waypoints = new List<float>();
00015             foreach (var arg in args)
00016             {
00017                 if (arg is float floatArg)
00018                 {
00019                     waypoints.Add(floatArg);
00020                 }
00021                 else if (arg is IEnumerable<float> floatEnumerable)
00022                 {
00023                     waypoints.AddRange(floatEnumerable);
00024                 }
00025             }
00026             return new WaypointCurveFloatAngleSeeker(Duration, ProgressCurve, waypoints);
00027         }
00028     }
00029 }
```

7.87 WaypointCurveFloatSeeker.cs

```
00001 using System.Collections.Generic;
00002 using System.Linq;
00003 using UnityEngine;
00004 
00005 namespace CardHouse
00006 {
00007     public class WaypointCurveFloatSeeker : AnimCurveFloatSeeker
00008     {
00009         List<float> Waypoints;
00010 
00011         public WaypointCurveFloatSeeker(float duration, AnimationCurve progressCurve, List<float>
    waypoints) : base(duration, progressCurve)
00012         {
```

```
00013             Waypoints = waypoints.ToList();
00014         }
00015
00016         public override Seeker<float> MakeCopy()
00017         {
00018             return new WaypointCurveFloatSeeker(Duration, ProgressCurve, Waypoints);
00019         }
00020
00021         public override float Pump(float currentValue, float TimeSinceLastFrame)
00022         {
00023             Timer += TimeSinceLastFrame;
00024             var normalizedTime = Timer / Duration;
00025             var progress = Mathf.Min(ProgressCurve.Evaluate(normalizedTime), 0.9999f);
00026             var destIndex = Mathf.FloorToInt(progress * (Waypoints.Count + 1));
00027             var dest = destIndex >= Waypoints.Count ? End : Waypoints[destIndex];
00028             var lastWaypoint = destIndex == 0 ? Start : Waypoints[destIndex - 1];
00029             return lastWaypoint + (dest - lastWaypoint) * (progress % (1f / (Waypoints.Count + 1))) *
      (Waypoints.Count + 1);
00030         }
00031
00032         public override bool IsDone(float currentValue)
00033         {
00034             return Timer >= Duration;
00035         }
00036     }
00037 }
```

7.88 WaypointCurveFloatSeekerScriptable.cs

```
00001 using System.Collections.Generic;
00002 using UnityEngine;
00003
00004 namespace CardHouse
00005 {
00006     [CreateAssetMenu(menuName = "CardHouse/Seekers/Float/Waypoint Curve")]
00007     public class WaypointCurveFloatSeekerScriptable : SeekerScriptable<float>
00008     {
00009         public float Duration = 2f;
00010         public AnimationCurve ProgressCurve;
00011
00012         public override Seeker<float> GetStrategy(params object[] args)
00013         {
00014             var waypoints = new List<float>();
00015             foreach (var arg in args)
00016             {
00017                 if (arg is float floatArg)
00018                 {
00019                     waypoints.Add(floatArg);
00020                 }
00021                 else if (arg is IEnumerable<float> floatEnumerable)
00022                 {
00023                     waypoints.AddRange(floatEnumerable);
00024                 }
00025             }
00026             return new WaypointCurveFloatSeeker(Duration, ProgressCurve, waypoints);
00027         }
00028     }
00029 }
```

7.89 Seeker.cs

```
00001 namespace CardHouse
00002 {
00003     public abstract class Seeker<T>
00004     {
00005         protected T Start;
00006         public T End;
00007
00008         public abstract Seeker<T> MakeCopy();
00009
00010         public void StartSeeking(T from, T to)
00011         {
00012             Start = from;
00013             End = to;
00014         }
00015
00016         public abstract T Pump(T currentValue, float TimeSinceLastFrame);
00017
00018         public abstract bool IsDone(T currentValue);
00019     }
00020 }
```

7.90 SeekerScriptable.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     public abstract class SeekerScriptable<T> : ScriptableObject
00006     {
00007         public abstract Seeker<T> GetStrategy(params object[] args);
00008     }
00009 }
```

7.91 SeekerScriptableSet.cs

```
00001 using System;
00002 using UnityEngine;
00003
00004 namespace CardHouse
00005 {
00006     [Serializable]
00007     public class SeekerScriptableSet
00008     {
00009         public SeekerScriptable<Vector3> Homing;
00010         public SeekerScriptable<float> Turning;
00011         public SeekerScriptable<float> Scaling;
00012     }
00013 }
```

7.92 SeekerSet.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     public class SeekerSet
00006     {
00007         public Card Card;
00008         public Seeker<Vector3> Homing;
00009         public Seeker<float> Turning;
00010         public Seeker<float> Scaling;
00011         public float FlipSpeed = 1f;
00012     }
00013 }
```

7.93 SeekerSetList.cs

```
00001 using System.Collections.Generic;
00002 using System.Linq;
00003
00004 namespace CardHouse
00005 {
00006     public class SeekerSetList : List<SeekerSet>
00007     {
00008         public SeekerSet GetSeekerSetFor(Card card)
00009         {
00010             foreach (var seekerSet in this)
00011             {
00012                 if (seekerSet == null)
00013                     continue;
00014
00015                 if (seekerSet.Card == card)
00016                 {
00017                     return seekerSet;
00018                 }
00019             }
00020
00021             return this.FirstOrDefault(x => x != null && x.Card == null);
00022         }
00023     }
00024 }
```

7.94 AnimCurveVector3Seeker.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     public class AnimCurveVector3Seeker : Seeker<Vector3>
00006     {
00007         public float Duration;
00008         protected float Timer;
00009         public AnimationCurve ProgressCurve;
00010
00011         public AnimCurveVector3Seeker(float duration, AnimationCurve progressCurve)
00012         {
00013             Duration = duration;
00014             ProgressCurve = progressCurve;
00015             Timer = 0f;
00016         }
00017
00018         public override Seeker<Vector3> MakeCopy()
00019         {
00020             return new AnimCurveVector3Seeker(Duration, ProgressCurve);
00021         }
00022
00023         public override Vector3 Pump(Vector3 currentValue, float TimeSinceLastFrame)
00024         {
00025             Timer += TimeSinceLastFrame;
00026             var normalizedTime = Timer / Duration;
00027             return Start + (End - Start) * ProgressCurve.Evaluate(normalizedTime);
00028         }
00029
00030         public override bool IsDone(Vector3 currentValue)
00031         {
00032             return Timer >= Duration;
00033         }
00034     }
00035 }
```

7.95 AnimCurveVector3SeekerScriptable.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     [CreateAssetMenu(menuName = "CardHouse/Seekers/Vector3/Animated Curve")]
00006     public class AnimCurveVector3SeekerScriptable : SeekerScriptable<Vector3>
00007     {
00008         public float Duration = 2f;
00009         public AnimationCurve ProgressCurve;
00010
00011         public override Seeker<Vector3> GetStrategy(params object[] args)
00012         {
00013             return new AnimCurveVector3Seeker(Duration, ProgressCurve);
00014         }
00015     }
00016 }
```

7.96 ContinuousInstantVector3Seeker.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     public class ContinuousInstantVector3Seeker : Seeker<Vector3>
00006     {
00007         public override Seeker<Vector3> MakeCopy()
00008         {
00009             return new ContinuousInstantVector3Seeker();
00010         }
00011
00012         public override Vector3 Pump(Vector3 currentValue, float TimeSinceLastFrame)
00013         {
00014             return End;
00015         }
00016
00017         public override bool IsDone(Vector3 currentValue)
00018         {
00019             return false;
00020         }
00021     }
00022 }
```

7.97 ContinuousInstantVector3SeekerScriptable.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     [CreateAssetMenu(menuName = "CardHouse/Seekers/Vector3/Continuous")]
00006     public class ContinuousInstantVector3SeekerScriptable : SeekerScriptable<Vector3>
00007     {
00008         public override Seeker<Vector3> GetStrategy(params object[] args)
00009         {
00010             return new ContinuousInstantVector3Seeker();
00011         }
00012     }
00013 }
```

7.98 ExponentialVector3Seeker.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     public class ExponentialVector3Seeker : Seeker<Vector3>
00006     {
00007         float XYGain = 8f;
00008         float ZGain = 3f; // Want to home Z faster than X and Y so that cards don't slide through each
      other as much
00009         float ArrivalDistance;
00010
00011         public ExponentialVector3Seeker(float xyGain = 8f, float zGain = 10f, float arrivalDist =
      0.01f)
00012         {
00013             XYGain = xyGain;
00014             ZGain = zGain;
00015             ArrivalDistance = arrivalDist;
00016         }
00017
00018         public override Seeker<Vector3> MakeCopy()
00019         {
00020             return new ExponentialVector3Seeker(XYGain, ZGain, ArrivalDistance);
00021         }
00022
00023         public override Vector3 Pump(Vector3 currentValue, float TimeSinceLastFrame)
00024         {
00025             return currentValue + (Vector3.right * (End.x - currentValue.x) + Vector3.up * (End.y -
      currentValue.y)) * XYGain * TimeSinceLastFrame + Vector3.forward * (End.z - currentValue.z) * ZGain *
      TimeSinceLastFrame;
00026         }
00027
00028         public override bool IsDone(Vector3 currentValue)
00029         {
00030             return (currentValue - End).magnitude <= ArrivalDistance;
00031         }
00032     }
00033 }
```

7.99 ExponentialVector3SeekerScriptable.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     [CreateAssetMenu(menuName = "CardHouse/Seekers/Vector3/Exponential")]
00006     public class ExponentialVector3SeekerScriptable : SeekerScriptable<Vector3>
00007     {
00008         public float XYGain = 8f;
00009         public float ZGain = 10f;
00010         public float ArrivalDistance = 0.01f;
00011
00012         public override Seeker<Vector3> GetStrategy(params object[] args)
00013         {
00014             return new ExponentialVector3Seeker(XYGain, ZGain, ArrivalDistance);
00015         }
00016     }
00017 }
```

7.100 InstantVector3Seeker.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     public class InstantVector3Seeker : Seeker<Vector3>
00006     {
00007         public override Seeker<Vector3> MakeCopy()
00008         {
00009             return new InstantVector3Seeker();
00010         }
00011
00012         public override Vector3 Pump(Vector3 currentValue, float TimeSinceLastFrame)
00013         {
00014             return End;
00015         }
00016
00017         public override bool IsDone(Vector3 currentValue)
00018         {
00019             return true;
00020         }
00021     }
00022 }
```

7.101 RandomizedCurveVector3SeekerScriptable.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     [CreateAssetMenu(menuName = "CardHouse/Seekers/Vector3/Randomized Curve")]
00006     public class RandomizedCurveVector3SeekerScriptable : TweakVector3SeekerScriptable
00007     {
00008         public float TweakMagnitudeMin = 1.5f;
00009         public float TweakMagnitudeMax = 2f;
00010
00011         public override Seeker<Vector3> GetStrategy(params object[] args)
00012         {
00013             var myAngle = Random.Range(0f, 360f);
00014             var tweak = Vector3.right * Mathf.Cos(myAngle) + Vector3.up * Mathf.Sin(myAngle);
00015             return new TweakVector3Seeker(Duration, ProgressCurve, Random.Range(TweakMagnitudeMin,
      TweakMagnitudeMax) * tweak, TweakMultiplier);
00016         }
00017     }
00018 }
```

7.102 TweakVector3Seeker.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     public class TweakVector3Seeker : AnimCurveVector3Seeker
00006     {
00007         public Vector3 Tweak;
00008         public AnimationCurve TweakMultiplier;
00009
00010         public TweakVector3Seeker(float duration, AnimationCurve progressCurve, Vector3 tweak,
      AnimationCurve tweakMultiplier) : base(duration, progressCurve)
00011         {
00012             Tweak = tweak;
00013             TweakMultiplier = tweakMultiplier;
00014         }
00015
00016         public override Seeker<Vector3> MakeCopy()
00017         {
00018             return new TweakVector3Seeker(Duration, ProgressCurve, Tweak, TweakMultiplier);
00019         }
00020
00021         public override Vector3 Pump(Vector3 currentValue, float TimeSinceLastFrame)
00022         {
00023             var step = base.Pump(currentValue, TimeSinceLastFrame);
00024             var normalizedTime = Timer / Duration;
00025             return step + TweakMultiplier.Evaluate(normalizedTime) * Tweak;
00026         }
00027     }
00028 }
```

7.103 TweakVector3SeekerScriptable.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     [CreateAssetMenu(menuName = "CardHouse/Seekers/Vector3/Tweak")]
00006     public class TweakVector3SeekerScriptable : AnimCurveVector3SeekerScriptable
00007     {
00008         public Vector3 Tweak;
00009         public AnimationCurve TweakMultiplier;
00010
00011         public override Seeker<Vector3> GetStrategy(params object[] args)
00012         {
00013             return new TweakVector3Seeker(Duration, ProgressCurve, Tweak, TweakMultiplier);
00014         }
00015     }
00016 }
```

7.104 WaypointCurveVector3Seeker.cs

```
00001 using System.Collections.Generic;
00002 using System.Linq;
00003 using UnityEngine;
00004
00005 namespace CardHouse
00006 {
00007     public class WaypointCurveVector3Seeker : AnimCurveVector3Seeker
00008     {
00009         List<Vector3> Waypoints;
00010
00011         public WaypointCurveVector3Seeker(float duration, AnimationCurve progressCurve, List<Vector3>
      waypoints) : base(duration, progressCurve)
00012         {
00013             Waypoints = waypoints.ToList();
00014         }
00015
00016         public override Seeker<Vector3> MakeCopy()
00017         {
00018             return new WaypointCurveVector3Seeker(Duration, ProgressCurve, Waypoints);
00019         }
00020
00021         public override Vector3 Pump(Vector3 currentValue, float TimeSinceLastFrame)
00022         {
00023             Timer += TimeSinceLastFrame;
00024             var normalizedTime = Timer / Duration;
00025             var progress = Mathf.Min(ProgressCurve.Evaluate(normalizedTime), 0.9999f);
00026             var destIndex = Mathf.FloorToInt(progress * (Waypoints.Count + 1));
00027             var dest = destIndex >= Waypoints.Count ? End : Waypoints[destIndex];
00028             var lastWaypoint = destIndex == 0 ? Start : Waypoints[destIndex - 1];
00029             return lastWaypoint + (dest - lastWaypoint) * (progress % (1f / (Waypoints.Count + 1))) *
      (Waypoints.Count + 1);
00030         }
00031
00032         public override bool IsDone(Vector3 currentValue)
00033         {
00034             return Timer >= Duration;
00035         }
00036     }
00037 }
```

7.105 WaypointCurveVector3SeekerScriptable.cs

```
00001 using System.Collections.Generic;
00002 using UnityEngine;
00003
00004 namespace CardHouse
00005 {
00006     [CreateAssetMenu(menuName = "CardHouse/Seekers/Vector3/Waypoint Curve")]
00007     public class WaypointCurveVector3SeekerScriptable : SeekerScriptable<Vector3>
00008     {
00009         public float Duration = 2f;
00010         public AnimationCurve ProgressCurve;
00011
00012         public override Seeker<Vector3> GetStrategy(params object[] args)
00013         {
00014             var waypoints = new List<Vector3>();
00015             foreach (var arg in args)
00016             {
```

```
00017                 if (arg is Vector3 v3)
00018                 {
00019                     waypoints.Add(v3);
00020                 }
00021                 else if (arg is IEnumerable<Vector3> v3Enumerable)
00022                 {
00023                     waypoints.AddRange(v3Enumerable);
00024                 }
00025             }
00026             return new WaypointCurveVector3Seeker(Duration, ProgressCurve, waypoints);
00027         }
00028     }
00029 }
```

7.106 CardSetup.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse
00004 {
00005     public abstract class CardSetup : MonoBehaviour
00006     {
00007         public abstract void Apply(CardDefinition data);
00008     }
00009 }
```

7.107 DeckSetup.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 namespace CardHouse
00006 {
00007     public class DeckSetup : MonoBehaviour
00008     {
00009         public bool RunOnStart = true;
00010         public GameObject CardPrefab;
00011         public DeckDefinition DeckDefinition;
00012         public CardGroup Deck;
00013         public List<TimedEvent> OnSetupCompleteEventChain;
00014
00015         void Start()
00016         {
00017             if (RunOnStart)
00018             {
00019                 DoSetup();
00020             }
00021         }
00022
00023         public void DoSetup()
00024         {
00025             StartCoroutine(SetupCoroutine());
00026         }
00027
00028         IEnumerator SetupCoroutine()
00029         {
00030             var newCardList = new List<Card>();
00031             foreach (var cardDef in DeckDefinition.CardCollection)
00032             {
00033                 var newThing = Instantiate(CardPrefab, Deck.transform.position,
      Deck.transform.rotation);
00034                 newCardList.Add(newThing.GetComponent<Card>());
00035                 var card = newThing.GetComponent<CardSetup>();
00036
00037                 if (card != null)
00038                 {
00039                     var copyCardDef = cardDef;
00040
00041                     if (cardDef.BackArt == null && DeckDefinition.CardBackArt != null)
00042                     {
00043                         copyCardDef = Instantiate(cardDef);
00044                         copyCardDef.BackArt = DeckDefinition.CardBackArt;
00045                     }
00046                     card.Apply(copyCardDef);
00047                 }
00048             }
00049
00050             yield return new WaitForEndOfFrame();
```

```
00051
00052             foreach (var card in newCardList)
00053             {
00054                 Deck.Mount(card, instaFlip: true);
00055             }
00056
00057             Deck.Shuffle();
00058
00059             StartCoroutine(TimedEvent.ExecuteChain(OnSetupCompleteEventChain));
00060         }
00061     }
00062 }
```

7.108 GroupSetup.cs

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using UnityEngine;
00005
00006 namespace CardHouse
00007 {
00008     public class GroupSetup : MonoBehaviour
00009     {
00010         [Serializable]
00011         public struct GroupPopulationData
00012         {
00013             public CardGroup Group;
00014             public GameObject CardPrefab;
00015             public int CardCount;
00016         }
00017
00018         public bool RunOnStart = true;
00019
00020         public List<GroupPopulationData> GroupPopulationList;
00021
00022         public List<CardGroup> GroupsToShuffle;
00023
00024         public List<TimedEvent> OnSetupCompleteEventChain;
00025
00026         void Start()
00027         {
00028             if (RunOnStart)
00029             {
00030                 DoSetup();
00031             }
00032         }
00033
00034         public void DoSetup()
00035         {
00036             StartCoroutine(SetupCoroutine());
00037         }
00038
00039         IEnumerator SetupCoroutine()
00040         {
00041             var homing = new InstantVector3Seeker();
00042             var turning = new InstantFloatSeeker();
00043             var newThingDict = new Dictionary<GroupPopulationData, List<GameObject»();
00044             foreach (var group in GroupPopulationList)
00045             {
00046                 newThingDict[group] = new List<GameObject>();
00047                 for (var i = 0; i < group.CardCount; i++)
00048                 {
00049                     var newThing = Instantiate(group.CardPrefab, Vector3.down * 10,
      Quaternion.identity);
00050                     newThingDict[group].Add(newThing);
00051                 }
00052             }
00053
00054             yield return new WaitForEndOfFrame();
00055
00056             foreach (var kvp in newThingDict)
00057             {
00058                 foreach (var newThing in kvp.Value)
00059                 {
00060                     kvp.Key.Group.Mount(newThing.GetComponent<Card>(), instaFlip: true, seekerSets:
      new SeekerSetList { new SeekerSet { Homing = homing, Turning = turning } });
00061                 }
00062             }
00063
00064             foreach (var group in GroupsToShuffle)
00065             {
00066                 group.Shuffle(true);
```

```
00067             }
00068
00069             StartCoroutine(TimedEvent.ExecuteChain(OnSetupCompleteEventChain));
00070         }
00071     }
00072 }
```

7.109 MultiplayerBoardSetup.cs

```
00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using UnityEngine;
00005 using UnityEngine.SceneManagement;
00006 using UnityEngine.UI;
00007
00008 namespace CardHouse
00009 {
00010     public class MultiplayerBoardSetup : MonoBehaviour
00011     {
00012         [Serializable]
00013         public class GroupTransitionByName
00014         {
00015             public int PhaseIndex;
00016             public GroupName Source;
00017             public GroupName Destination;
00018             public DragAction DragAction;
00019             public PvpMode Mode;
00020         }
00021
00022         public enum PvpMode
00023         {
00024             PlayerToEnemy,
00025             EnemyToPlayer
00026         }
00027
00028         public bool RunOnStart = true;
00029         public GameObject PlayerBoard;
00030         public int PlayerCount = 2;
00031         public float SpacingMultiplier = 1.0f;
00032
00033         public List<GroupTransitionByName> PlayerToPlayerInteractions;
00034
00035         List<GameObject> SpawnedBoards = new List<GameObject>();
00036         List<GroupSetup> SpawnedGroupSetups = new List<GroupSetup>();
00037         List<DeckSetup> SpawnedDeckSetups = new List<DeckSetup>();
00038         List<Phase> SpawnedPhases = new List<Phase>();
00039         Dictionary<int, List<DragTransition>» PvpDragTransitionsAddedToTemplate = new Dictionary<int,
      List<DragTransition»();
00040
00041         Vector3 Size;
00042         Vector3 Offset;
00043
00044         public GameObject[] GetAllBoards()
00045         {
00046             var boards = new List<GameObject> { PlayerBoard };
00047             boards.AddRange(SpawnedBoards);
00048             return boards.ToArray();
00049         }
00050
00051         private void Start()
00052         {
00053             if (RunOnStart)
00054             {
00055                 Setup(false);
00056             }
00057         }
00058
00059         public void Setup(bool callSetupScripts = true)
00060         {
00061             if (PlayerCount <= 0)
00062                 return;
00063
00064             Teardown();
00065
00066             // Find bounds
00067             var bottom = 0f;
00068             var top = 0f;
00069             var left = 0f;
00070             var right = 0f;
00071             foreach (var cardGroup in PlayerBoard.GetComponentsInChildren<CardGroup>())
00072             {
00073                 left = Mathf.Min(left, cardGroup.transform.position.x -
      cardGroup.transform.lossyScale.x / 2f);
```

```
00074                 right = Mathf.Max(right, cardGroup.transform.position.x +
      cardGroup.transform.lossyScale.x / 2f);
00075                 bottom = Mathf.Min(bottom, cardGroup.transform.position.y -
      cardGroup.transform.lossyScale.y / 2f);
00076                 top = Mathf.Max(top, cardGroup.transform.position.y + cardGroup.transform.lossyScale.y
      / 2f);
00077             }
00078
00079             Size = new Vector3(right - left, top - bottom, 0);
00080             Offset = new Vector3(left + Size.x / 2f - PlayerBoard.transform.position.x, bottom +
      Size.y / 2f - PlayerBoard.transform.position.y, 0);
00081
00082             // Analyze template board
00083             var originalBoardGroups = PlayerBoard.GetComponentsInChildren<CardGroup>();
00084             var originalBoardButtons = PlayerBoard.GetComponentsInChildren<Button>();
00085             var originalBoardClickables = PlayerBoard.GetComponentsInChildren<ClickDetector>();
00086             var originalBoardComponents = PlayerBoard.GetComponentsInChildren<MonoBehaviour>();
00087             var originalBoardGroupSetups = PlayerBoard.GetComponentsInChildren<GroupSetup>();
00088             var originalBoardDeckSetups = PlayerBoard.GetComponentsInChildren<DeckSetup>();
00089
00090             // Find external setup scripts
00091             var oddGroupSetups = SceneManager.GetActiveScene().GetRootGameObjects().SelectMany(x =>
      x.GetComponentsInChildren<GroupSetup>()).Where(x => !originalBoardGroupSetups.Contains(x)).ToArray();
00092             var oddDeckSetups = SceneManager.GetActiveScene().GetRootGameObjects().SelectMany(x =>
      x.GetComponentsInChildren<DeckSetup>()).Where(x => !originalBoardDeckSetups.Contains(x)).ToArray();
00093
00094             // Find external card groups
00095             var oddGroups = SceneManager.GetActiveScene().GetRootGameObjects().SelectMany(x =>
      x.GetComponentsInChildren<CardGroup>()).Where(x => !originalBoardGroups.Contains(x)).ToArray();
00096
00097             if (callSetupScripts)
00098             {
00099                 foreach (var setup in originalBoardGroupSetups)
00100                 {
00101                     setup.DoSetup();
00102                 }
00103                 foreach (var setup in oddGroupSetups)
00104                 {
00105                     setup.DoSetup();
00106                 }
00107                 foreach (var setup in originalBoardDeckSetups)
00108                 {
00109                     setup.DoSetup();
00110                 }
00111                 foreach (var setup in oddDeckSetups)
00112                 {
00113                     setup.DoSetup();
00114                 }
00115             }
00116
00117             // Analyze Group Registry
00118             var registeredGroups = new Dictionary<GroupName, CardGroup>();
00119             if (GroupRegistry.Instance != null)
00120             {
00121                 foreach (var group in GroupRegistry.Instance.Groups)
00122                 {
00123                     if (group.PlayerIndex == 0)
00124                     {
00125                         registeredGroups.Add(group.Name, group.Group);
00126                     }
00127                 }
00128             }
00129
00130             // Setup boards
00131             var newPhases = new Dictionary<int, List<Phase>>();
00132
00133             var marginalAngle = 360f / PlayerCount;
00134             var distanceToCenter = Size.y * 110f * SpacingMultiplier / marginalAngle;
00135             var centerOfCircle = PlayerBoard.transform.position + Offset + Vector3.up *
      distanceToCenter;
00136
00137             // Scale camera and camera points
00138             Camera.main.orthographicSize = distanceToCenter + Size.y / 2f;
00139             foreach (var phase in PhaseManager.Instance.Phases)
00140             {
00141                 if (phase.CameraPosition == null)
00142                     continue;
00143
00144                 phase.CameraPosition.localPosition = Offset + Vector3.up * (distanceToCenter);
00145             }
00146
00147             for (var i = 1; i < PlayerCount; i++)
00148             {
00149                 // Duplicate and space
00150                 var newBoard = Instantiate(PlayerBoard.gameObject);
00151                 SpawnedBoards.Add(newBoard);
00152                 newBoard.transform.RotateAround(centerOfCircle, Vector3.forward, marginalAngle * i);
```

```
00153
00154                var boardGroups = newBoard.GetComponentsInChildren<CardGroup>();
00155                var boardTransforms = newBoard.GetComponentsInChildren<Transform>();
00156                var boardButtons = newBoard.GetComponentsInChildren<Button>();
00157                var boardClickables = newBoard.GetComponentsInChildren<ClickDetector>();
00158                var boardComponents = newBoard.GetComponentsInChildren<MonoBehaviour>();
00159
00160                // Group Registry
00161                if (GroupRegistry.Instance != null)
00162                {
00163                    foreach (var registeredGroup in registeredGroups)
00164                    {
00165                        var correspondingGroup =
     boardGroups.GetComponentForName(registeredGroup.Value.gameObject.name);
00166                        if (correspondingGroup != null)
00167                        {
00168                            GroupRegistry.Instance.Groups.Add(new GroupRegistry.NamedGroup { Name =
     registeredGroup.Key, PlayerIndex = i, Group = correspondingGroup });
00169                        }
00170                    }
00171                }
00172
00173                // Group Setup
00174                foreach (var groupSetup in oddGroupSetups)
00175                {
00176                    var groupSetupEntriesToAdd = new List<GroupSetup.GroupPopulationData>();
00177
00178                    foreach (var target in groupSetup.GroupPopulationList)
00179                    {
00180                        if (originalBoardGroups.Contains(target.Group))
00181                        {
00182                            var correspondingGroup =
     boardGroups.GetComponentForName(target.Group.gameObject.name);
00183                            groupSetupEntriesToAdd.Add(new GroupSetup.GroupPopulationData { Group =
     correspondingGroup, CardPrefab = target.CardPrefab, CardCount = target.CardCount });
00184                        }
00185                    }
00186
00187                    var newGroupSetup = groupSetup.gameObject.AddComponent<GroupSetup>();
00188                    SpawnedGroupSetups.Add(newGroupSetup);
00189                    newGroupSetup.RunOnStart = groupSetup.RunOnStart;
00190                    newGroupSetup.GroupPopulationList = groupSetupEntriesToAdd;
00191                    newGroupSetup.GroupsToShuffle = new List<CardGroup>();
00192
00193                    foreach (var group in groupSetup.GroupsToShuffle.ToArray())
00194                    {
00195                        if (originalBoardGroups.Contains(group))
00196                        {
00197                            var correspondingGroup =
     boardGroups.GetComponentForName(group.gameObject.name);
00198                            newGroupSetup.GroupsToShuffle.Add(correspondingGroup);
00199                        }
00200                    }
00201
00202                    newGroupSetup.OnSetupCompleteEventChain =
     CopyEvents(groupSetup.OnSetupCompleteEventChain, originalBoardComponents, boardComponents);
00203                    if (callSetupScripts)
00204                    {
00205                        newGroupSetup.DoSetup();
00206                    }
00207                }
00208
00209                // Deck Setup
00210                foreach (var deckSetup in oddDeckSetups)
00211                {
00212                    if (originalBoardGroups.Contains(deckSetup.Deck))
00213                    {
00214                        var correspondingGroup =
     boardGroups.GetComponentForName(deckSetup.Deck.gameObject.name);
00215                        var newDeckSetup = deckSetup.gameObject.AddComponent<DeckSetup>();
00216                        SpawnedDeckSetups.Add(newDeckSetup);
00217                        newDeckSetup.RunOnStart = deckSetup.RunOnStart;
00218                        newDeckSetup.Deck = correspondingGroup;
00219                        newDeckSetup.CardPrefab = deckSetup.CardPrefab;
00220                        newDeckSetup.DeckDefinition = deckSetup.DeckDefinition;
00221                        newDeckSetup.OnSetupCompleteEventChain =
     CopyEvents(deckSetup.OnSetupCompleteEventChain, originalBoardComponents, boardComponents);
00222                        if (callSetupScripts)
00223                        {
00224                            newDeckSetup.DoSetup();
00225                        }
00226                    }
00227                }
00228
00229                // Resource Manager
00230                if (CurrencyRegistry.Instance != null && CurrencyRegistry.Instance.PlayerWallets.Count
     > 0)
```

```
00231                 {
00232
      CurrencyRegistry.Instance.PlayerWallets.Add((CurrencyWallet)CurrencyRegistry.Instance.PlayerWallets[0].Clone());
00233                 }
00234
00235                 // Phase Manager
00236                 if (PhaseManager.Instance != null)
00237                 {
00238                     var p1Phases = PhaseManager.Instance.Phases.Where(x => x.PlayerIndex ==
      0).Select(x => PhaseManager.Instance.Phases.IndexOf(x)).ToArray();
00239                     foreach (var phaseIndex in p1Phases)
00240                     {
00241                         var phase = PhaseManager.Instance.Phases[phaseIndex];
00242                         var newPhase = new Phase
00243                         {
00244                             Name = phase.Name.Replace("1", (i + 1).ToString()),
00245                             PlayerIndex = i,
00246                             CameraPosition =
      boardTransforms.GetComponentForName(phase.CameraPosition.gameObject.name),
00247                             CardPresentationPosition =
      boardTransforms.GetComponentForName(phase.CardPresentationPosition.gameObject.name),
00248                         };
00249                         SpawnedPhases.Add(newPhase);
00250
00251                         newPhase.ActiveButtons = phase.ActiveButtons.ToList();
00252                         for (var j = 0; j < newPhase.ActiveButtons.Count; j++)
00253                         {
00254                             if (originalBoardButtons.Contains(newPhase.ActiveButtons[j]))
00255                             {
00256                                 newPhase.ActiveButtons[j] =
      boardButtons.GetComponentForName(newPhase.ActiveButtons[j].gameObject.name);
00257                             }
00258                         }
00259
00260                         newPhase.ValidClickTargets = phase.ValidClickTargets.ToList();
00261                         for (var j = 0; j < newPhase.ValidClickTargets.Count; j++)
00262                         {
00263                             if (originalBoardClickables.Contains(newPhase.ValidClickTargets[j]))
00264                             {
00265                                 newPhase.ValidClickTargets[j] =
      boardClickables.GetComponentForName(newPhase.ValidClickTargets[j].gameObject.name);
00266                             }
00267                         }
00268
00269                         //  Valid Drags
00270                         newPhase.ValidDrags = phase.ValidDrags.Select(x => new DragTransition { Source
      = x.Source, Destination = x.Destination, DragAction = x.DragAction }).ToList();
00271                         foreach (var drag in newPhase.ValidDrags)
00272                         {
00273                             if (originalBoardGroups.Contains(drag.Source))
00274                             {
00275                                 drag.Source =
      boardGroups.GetComponentForName(drag.Source.gameObject.name);
00276                             }
00277
00278                             if (originalBoardGroups.Contains(drag.Destination))
00279                             {
00280                                 drag.Destination =
      boardGroups.GetComponentForName(drag.Destination.gameObject.name);
00281                             }
00282                         }
00283
00284                         //  Beginning/End of phase events
00285                         newPhase.OnPhaseStartEventChain = CopyEvents(phase.OnPhaseStartEventChain,
      originalBoardComponents, boardComponents);
00286                         newPhase.OnPhaseEndEventChain = CopyEvents(phase.OnPhaseEndEventChain,
      originalBoardComponents, boardComponents);
00287
00288                         if (newPhases.ContainsKey(phaseIndex))
00289                         {
00290                             newPhases[phaseIndex].Add(newPhase);
00291                         }
00292                         else
00293                         {
00294                             newPhases[phaseIndex] = new List<Phase> { newPhase };
00295                         }
00296                     }
00297                 }
00298             }
00299
00300             var phasesByIndex = new Dictionary<int, List<Phase»();
00301             var reversedKeys = newPhases.Keys.ToList();
00302             reversedKeys.Sort();
00303             reversedKeys.Reverse();
00304             foreach (var phaseIndex in reversedKeys)
00305             {
00306                 PhaseManager.Instance.Phases.InsertRange(phaseIndex + 1, newPhases[phaseIndex]);
```

```
00307                 phasesByIndex[phaseIndex] = PhaseManager.Instance.Phases.GetRange(phaseIndex,
      newPhases[phaseIndex].Count + 1);
00308             }
00309
00310             // Set up player-to-player interactions
00311             reversedKeys.Reverse();
00312             foreach (var pvp in PlayerToPlayerInteractions)
00313             {
00314                 if (phasesByIndex.ContainsKey(pvp.PhaseIndex))
00315                 {
00316                     foreach (var phase in phasesByIndex[pvp.PhaseIndex])
00317                     {
00318                         for (var i = 0; i < PlayerCount; i++)
00319                         {
00320                             if (i == phase.PlayerIndex)
00321                                 continue;
00322
00323                             var sourceIndex = pvp.Mode == PvpMode.PlayerToEnemy ? phase.PlayerIndex :
      i;
00324                             var destinationIndex = pvp.Mode == PvpMode.PlayerToEnemy ? i :
      phase.PlayerIndex;
00325                             var newDragTransition = new DragTransition
00326                             {
00327                                 Source = GroupRegistry.Instance.Get(pvp.Source, sourceIndex),
00328                                 Destination = GroupRegistry.Instance.Get(pvp.Destination,
      destinationIndex),
00329                                 DragAction = pvp.DragAction
00330                             };
00331                             phase.ValidDrags.Add(newDragTransition);
00332
00333                             if (phase == phasesByIndex[pvp.PhaseIndex][0])
00334                             {
00335                                 if (PvpDragTransitionsAddedToTemplate.ContainsKey(pvp.PhaseIndex))
00336                                 {
00337
      PvpDragTransitionsAddedToTemplate[pvp.PhaseIndex].Add(newDragTransition);
00338                                 }
00339                                 else
00340                                 {
00341                                     PvpDragTransitionsAddedToTemplate.Add(pvp.PhaseIndex, new
      List<DragTransition> { newDragTransition });
00342                                 }
00343                             }
00344                         }
00345                     }
00346                 }
00347             }
00348
00349             // Move odd groups to center
00350             foreach (var group in oddGroups)
00351             {
00352                 group.gameObject.transform.Translate(centerOfCircle -
      oddGroups[0].transform.position);
00353             }
00354
00355             // Scale presentation points
00356             foreach (var phase in PhaseManager.Instance.Phases)
00357             {
00358                 if (phase.CardPresentationPosition == null)
00359                     continue;
00360
00361                 phase.CardPresentationPosition.localScale = Vector3.one * 1.5f *
      Camera.main.orthographicSize / 4f;
00362             }
00363         }
00364
00365         List<TimedEvent> CopyEvents(IEnumerable<TimedEvent> source, IEnumerable<MonoBehaviour>
      sourceComponents, IEnumerable<MonoBehaviour> destinationComponents)
00366         {
00367             var output = new List<TimedEvent>();
00368             foreach (var timedEvent in source)
00369             {
00370                 var newTimedEvent = new TimedEvent { Duration = timedEvent.Duration, Event = new
      UnityEngine.Events.UnityEvent() };
00371                 for (var j = 0; j < timedEvent.Event.GetPersistentEventCount(); j++)
00372                 {
00373                     var target = (MonoBehaviour)timedEvent.Event.GetPersistentTarget(j);
00374                     if (sourceComponents.Contains(target))
00375                     {
00376                         target = destinationComponents.GetComponentForName(target.gameObject.name,
      target.GetType());
00377                     }
00378
00379                     var methodName = timedEvent.Event.GetPersistentMethodName(j);
00380                     newTimedEvent.Event.AddListener(new UnityEngine.Events.UnityAction(() =>
      target.Invoke(methodName, 0f)));
00381                 }
```

```
00382
00383                 output.Add(newTimedEvent);
00384             }
00385             return output;
00386         }
00387
00388         void Teardown()
00389         {
00390             foreach (var group in SceneManager.GetActiveScene().GetRootGameObjects().SelectMany(x =>
      x.GetComponentsInChildren<CardGroup>()))
00391             {
00392                 foreach (var card in group.MountedCards.ToArray())
00393                 {
00394                     group.MountedCards.Remove(card);
00395                     Destroy(card.gameObject);
00396                 }
00397             }
00398
00399             foreach (var board in SpawnedBoards)
00400             {
00401                 Destroy(board.gameObject);
00402             }
00403             SpawnedBoards.Clear();
00404
00405             if (GroupRegistry.Instance != null)
00406             {
00407                 foreach (var group in GroupRegistry.Instance.Groups.ToArray())
00408                 {
00409                     if (group.PlayerIndex > 0)
00410                     {
00411                         GroupRegistry.Instance.Groups.Remove(group);
00412                     }
00413                 }
00414             }
00415
00416             if (CurrencyRegistry.Instance != null && CurrencyRegistry.Instance.PlayerWallets.Count >
      0)
00417             {
00418                 CurrencyRegistry.Instance.PlayerWallets = new List<CurrencyWallet> {
      CurrencyRegistry.Instance.PlayerWallets[0] };
00419             }
00420
00421             foreach (var setup in SpawnedGroupSetups)
00422             {
00423                 DestroyImmediate(setup);
00424             }
00425             SpawnedGroupSetups.Clear();
00426
00427             foreach (var setup in SpawnedDeckSetups)
00428             {
00429                 DestroyImmediate(setup);
00430             }
00431             SpawnedDeckSetups.Clear();
00432
00433             if (PhaseManager.Instance != null)
00434             {
00435                 foreach (var phase in SpawnedPhases)
00436                 {
00437                     PhaseManager.Instance.Phases.Remove(phase);
00438                 }
00439                 SpawnedPhases.Clear();
00440
00441                 foreach (var phaseIndex in PvpDragTransitionsAddedToTemplate.Keys)
00442                 {
00443                     foreach (var transition in PvpDragTransitionsAddedToTemplate[phaseIndex])
00444                     {
00445                         PhaseManager.Instance.Phases[phaseIndex].ValidDrags.Remove(transition);
00446                     }
00447                 }
00448                 PvpDragTransitionsAddedToTemplate.Clear();
00449
00450                 PhaseManager.Instance.HardReset();
00451             }
00452         }
00453
00454     }
00455 }
```

7.110 MultiSpriteOperator.cs

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

```
00004 namespace CardHouse
00005 {
00006     public class MultiSpriteOperator : MonoBehaviour
00007     {
00008         public List<SpriteOperator> SpriteOperators;
00009
00010         public void Activate(string name)
00011         {
00012             Activate(name, this);
00013         }
00014
00015         public void Activate(string name, Object voter)
00016         {
00017             foreach (var handler in SpriteOperators)
00018             {
00019                 handler?.Activate(name, voter);
00020             }
00021         }
00022
00023         public void Remove(Object voter)
00024         {
00025             foreach (var handler in SpriteOperators)
00026             {
00027                 handler?.Remove(voter);
00028             }
00029         }
00030
00031         public void RemoveVote()
00032         {
00033             foreach (var handler in SpriteOperators)
00034             {
00035                 handler?.Remove(this);
00036             }
00037         }
00038     }
00039 }
```

7.111 SpriteColorOperator.cs

```
00001 using System;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 namespace CardHouse
00006 {
00007     public class SpriteColorOperator : SpriteOperator
00008     {
00009         [Serializable]
00010         public class NamedColor
00011         {
00012             public string Name;
00013             public Color Color;
00014         }
00015
00016         public List<NamedColor> Colors;
00017
00018         protected override void ChangeSprite(string name)
00019         {
00020             foreach (var namedColor in Colors)
00021             {
00022                 if (namedColor.Name == name)
00023                 {
00024                     SpriteTarget.color = namedColor.Color;
00025                     break;
00026                 }
00027             }
00028         }
00029     }
00030 }
```

7.112 SpriteImageOperator.cs

```
00001 using System;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 namespace CardHouse
00006 {
00007     public class SpriteImageOperator : SpriteOperator
```

```
00008     {
00009         [Serializable]
00010         public class NamedSprite
00011         {
00012             public string Name;
00013             public Sprite Sprite;
00014         }
00015 
00016         public List<NamedSprite> Sprites;
00017 
00018         protected override void ChangeSprite(string name)
00019         {
00020             foreach (var sprite in Sprites)
00021             {
00022                 if (sprite.Name == name)
00023                 {
00024                     SpriteTarget.sprite = sprite.Sprite;
00025                     break;
00026                 }
00027             }
00028         }
00029     }
00030 }
```

7.113 SpriteOperator.cs

```
00001 using System.Collections.Generic;
00002 using System.Linq;
00003 using UnityEngine;
00004 
00005 namespace CardHouse
00006 {
00007     [RequireComponent(typeof(SpriteRenderer))]
00008     public abstract class SpriteOperator : MonoBehaviour
00009     {
00010         public string FavoredState;
00011         protected SpriteRenderer SpriteTarget;
00012         Dictionary<Object, string> Votes = new Dictionary<Object, string>();
00013 
00014         void Awake()
00015         {
00016             SpriteTarget = GetComponent<SpriteRenderer>();
00017         }
00018 
00019         public void Activate(string name)
00020         {
00021             Activate(name, this);
00022         }
00023 
00024         public void Activate(string name, Object voter)
00025         {
00026             Votes[voter] = name;
00027 
00028             if (SpriteTarget == null)
00029                 return;
00030 
00031             UpdateState();
00032         }
00033 
00034         public void Remove(Object voter)
00035         {
00036             Votes.Remove(voter);
00037 
00038             UpdateState();
00039         }
00040 
00041         void UpdateState()
00042         {
00043             var allVotes = Votes.Values.ToList();
00044             if (allVotes.Contains(FavoredState) || allVotes.Count == 0)
00045             {
00046                 ChangeSprite(FavoredState);
00047             }
00048             else if (AllSame(allVotes))
00049             {
00050                 ChangeSprite(allVotes[0]);
00051             }
00052         }
00053 
00054         bool AllSame(List<string> stringList)
00055         {
00056             var counts = new Dictionary<string, int>();
00057             foreach (var item in stringList)
```

```
00058             {
00059                 if (!counts.ContainsKey(item))
00060                 {
00061                     counts[item] = 1;
00062                 }
00063                 else
00064                 {
00065                     counts[item] += 1;
00066                 }
00067             }
00068 
00069             return counts.Count == 1;
00070         }
00071 
00072         protected abstract void ChangeSprite(string name);
00073     }
00074 }
```

7.114 Toggleable.cs

```
00001 using UnityEngine;
00002 
00003 namespace CardHouse
00004 {
00005     public class Toggleable : MonoBehaviour
00006     {
00007         public bool IsActive = true;
00008 
00009         public void SetIsActive(bool newValue)
00010         {
00011             IsActive = newValue;
00012         }
00013     }
00014 }
```

7.115 Utils.cs

```
00001 using System;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using UnityEngine;
00005 
00006 namespace CardHouse
00007 {
00008     public static class Utils
00009     {
00010         public static float CorrectAngle(float angle) // yields an angle beween -180 and 180
00011         {
00012             while (angle < 0)
00013             {
00014                 angle += 360;
00015             }
00016             while (angle > 360)
00017             {
00018                 angle -= 360;
00019             }
00020             return angle;
00021         }
00022 
00023         public static T CopyComponent<T>(T original, GameObject destination) where T : Component
00024         {
00025             System.Type type = original.GetType();
00026             Component copy = destination.AddComponent(type);
00027             System.Reflection.FieldInfo[] fields = type.GetFields();
00028             foreach (System.Reflection.FieldInfo field in fields)
00029             {
00030                 field.SetValue(copy, field.GetValue(original));
00031             }
00032             return copy as T;
00033         }
00034 
00035         public static T GetComponentForName<T>(this IEnumerable<T> list, string name) where T :
      Component
00036         {
00037             return list.FirstOrDefault(x => x.gameObject.name == name);
00038         }
00039 
00040         public static T GetComponentForName<T>(this IEnumerable<T> list, string name, Type searchType)
      where T : Component
```

```
00041         {
00042             return list.FirstOrDefault(x => x.gameObject.name == name && x.GetType() == searchType);
00043         }
00044     }
00045 }
```

7.116 DamageGroupOperator.cs

```
00001 namespace CardHouse.SampleGames.DeckBuilder
00002 {
00003     public class DamageGroupOperator : CardTargetCardOperator
00004     {
00005         public int Damage;
00006 
00007         protected override void ActOnTarget()
00008         {
00009             foreach (var target in Target.Group.MountedCards.ToArray())
00010             {
00011                 var health = target.GetComponent<Health>();
00012                 if (health == null)
00013                     return;
00014 
00015                 health.Change(-1 * Damage);
00016             }
00017         }
00018     }
00019 }
```

7.117 DamageTargetOperator.cs

```
00001 namespace CardHouse.SampleGames.DeckBuilder
00002 {
00003     public class DamageTargetOperator : CardTargetCardOperator
00004     {
00005         public int Damage;
00006 
00007         protected override void ActOnTarget()
00008         {
00009             var health = Target.GetComponent<Health>();
00010             if (health == null)
00011                 return;
00012 
00013             health.Change(-1 * Damage);
00014         }
00015     }
00016 }
```

7.118 Health.cs

```
00001 using TMPro;
00002 using UnityEngine;
00003 using UnityEngine.Events;
00004 
00005 namespace CardHouse.SampleGames.DeckBuilder
00006 {
00007     public class Health : MonoBehaviour
00008     {
00009         public TextMeshPro HealthText;
00010         public int HealthLevel;
00011         public UnityEvent OnDeath;
00012 
00013         void Start()
00014         {
00015             UpdateHealthText();
00016         }
00017 
00018         void UpdateHealthText()
00019         {
00020             HealthText.text = HealthLevel.ToString();
00021         }
00022 
00023         public void Change(int diff)
00024         {
00025             HealthLevel += diff;
00026             UpdateHealthText();
```

```
00027
00028             if (HealthLevel <= 0)
00029             {
00030                 OnDeath.Invoke();
00031             }
00032         }
00033     }
00034 }
```

7.119 MemoryCard.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse.SampleGames.MemoryMatch
00004 {
00005     public class MemoryCard : MonoBehaviour
00006     {
00007         public SpriteRenderer MySpriteRenderer;
00008         [HideInInspector]
00009         public Sprite MySprite;
00010
00011         MemoryGame MyGame;
00012
00013         void Start()
00014         {
00015             MyGame = MemoryGame.Instance;
00016             if (MyGame == null)
00017             {
00018                 Debug.LogError("MemoryCards need MemoryGame component to exist in scene!");
00019                 return;
00020             }
00021         }
00022
00023         public void Apply(Sprite sprite)
00024         {
00025             MySprite = sprite;
00026             MySpriteRenderer.sprite = sprite;
00027         }
00028
00029         public void OnFlippedUp()
00030         {
00031             MyGame.Flip(this);
00032         }
00033     }
00034 }
```

7.120 MemoryGame.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using UnityEngine;
00005
00006 namespace CardHouse.SampleGames.MemoryMatch
00007 {
00008     public class MemoryGame : MonoBehaviour
00009     {
00010         public GameObject CardPrefab;
00011         public List<CardGroup> Slots;
00012         public List<Sprite> Sprites;
00013
00014         public MemoryUI MyUI;
00015         public GameObject MatchEffect;
00016
00017         int Matches = 0;
00018         float Timer;
00019         bool IsTimerRunning = true;
00020
00021         MemoryCard FlippedCard;
00022
00023         public static MemoryGame Instance;
00024
00025         void Start()
00026         {
00027             Restart();
00028         }
00029
00030         public void Restart()
00031         {
```

```
00032             foreach (var slot in Slots)
00033             {
00034                 Timer = 0;
00035                 Matches = 0;
00036                 foreach (var card in slot.MountedCards.ToList())
00037                 {
00038                     slot.UnMount(card);
00039                     Destroy(card.gameObject);
00040                 }
00041             }
00042 
00043             var spritePool = Sprites.ToList();
00044             var cards = new List<Card>();
00045             for (var i = 0; i < Slots.Count / 2; i++)
00046             {
00047                 if (spritePool.Count == 0)
00048                     break;
00049 
00050                 var sprite = spritePool[Random.Range(0, spritePool.Count)];
00051                 spritePool.Remove(sprite);
00052                 for (var j = 0; j < 2; j++)
00053                 {
00054                     var newCard = Instantiate(CardPrefab);
00055                     cards.Add(newCard.GetComponent<Card>());
00056                     var artHandler = newCard.GetComponent<MemoryCard>();
00057                     if (artHandler != null)
00058                     {
00059                         artHandler.Apply(sprite);
00060                     }
00061                 }
00062             }
00063 
00064             var slotPool = Slots.ToList();
00065 
00066             for (var i = 0; i < cards.Count; i++)
00067             {
00068                 cards[i].SetFacing(CardFacing.FaceDown, immediate: true);
00069                 var slotIndex = Random.Range(0, slotPool.Count);
00070                 slotPool[slotIndex].Mount(cards[i], seekerSets: new SeekerSetList { new SeekerSet {
      Card = cards[i], Homing = new InstantVector3Seeker(), Turning = new InstantFloatSeeker() } });
00071                 slotPool.RemoveAt(slotIndex);
00072             }
00073 
00074             MyUI.UpdateMatches(Matches);
00075         }
00076 
00077         void Awake()
00078         {
00079             Instance = this;
00080         }
00081 
00082         void Update()
00083         {
00084             if (IsTimerRunning)
00085             {
00086                 Timer += Time.deltaTime;
00087                 MyUI.UpdateTimer(Timer);
00088             }
00089         }
00090 
00091         public void Flip(MemoryCard card)
00092         {
00093             if (card == FlippedCard)
00094                 return;
00095 
00096             if (FlippedCard == null)
00097             {
00098                 FlippedCard = card;
00099                 return;
00100             }
00101 
00102             if (card.MySprite == FlippedCard.MySprite)
00103             {
00104                 Matches++;
00105                 if (Matches >= Slots.Count / 2)
00106                 {
00107                     IsTimerRunning = false;
00108                 }
00109                 Instantiate(MatchEffect, card.transform.position + Vector3.back,
      card.transform.rotation);
00110                 Instantiate(MatchEffect, FlippedCard.transform.position + Vector3.back,
      FlippedCard.transform.rotation);
00111                 MyUI.UpdateMatches(Matches);
00112             }
00113             else
00114             {
00115                 StartCoroutine(FlipDownAfter(1f, FlippedCard, card));
```

```
00116             }
00117             FlippedCard = null;
00118         }
00119
00120         IEnumerator FlipDownAfter(float delay, MemoryCard card1, MemoryCard card2)
00121         {
00122             yield return new WaitForSeconds(delay);
00123             if (card1 != FlippedCard)
00124             {
00125                 card1.GetComponent<Card>().SetFacing(CardFacing.FaceDown);
00126             }
00127             if (card2 != FlippedCard)
00128             {
00129                 card2.GetComponent<Card>().SetFacing(CardFacing.FaceDown);
00130             }
00131         }
00132     }
00133 }
```

7.121 MemoryUI.cs

```
00001 using UnityEngine;
00002 using UnityEngine.UI;
00003
00004 namespace CardHouse.SampleGames.MemoryMatch
00005 {
00006     public class MemoryUI : MonoBehaviour
00007     {
00008         public Text TimerText;
00009         public Text MatchText;
00010
00011         public void UpdateMatches(int matches)
00012         {
00013             MatchText.text = string.Format("Matches: {0}", matches);
00014         }
00015
00016         public void UpdateTimer(float timer)
00017         {
00018             TimerText.text = string.Format("Time: {0:F0}", timer);
00019         }
00020     }
00021 }
```

7.122 SolitaireCardDragHandler.cs

```
00001 using System.Collections.Generic;
00002 using UnityEngine;
00003
00004 namespace CardHouse.SampleGames.Solitaire
00005 {
00006     [RequireComponent(typeof(Card))]
00007     public class SolitaireCardDragHandler : MonoBehaviour
00008     {
00009         Card MyCard;
00010
00011         List<Transform> MyChildren = new List<Transform>();
00012
00013         void Awake()
00014         {
00015             MyCard = GetComponent<Card>();
00016         }
00017
00018         public void AttachChildren()
00019         {
00020             MyChildren.Clear();
00021             for (var i = MyCard.Group.MountedCards.IndexOf(MyCard) + 1; i <
      MyCard.Group.MountedCards.Count; i++)
00022             {
00023                 var childTransform = MyCard.Group.MountedCards[i].transform;
00024                 childTransform.parent = MyCard.transform;
00025                 MyChildren.Add(childTransform);
00026             }
00027         }
00028
00029         public void DetatchChildren()
00030         {
00031             foreach (var child in MyChildren)
00032             {
00033                 child.parent = null;
```

```
00034                 var childCard = child.GetComponent<Card>();
00035                 if (childCard.Group != MyCard.Group)
00036                 {
00037                     MyCard.Group.Mount(childCard);
00038                 }
00039             }
00040             MyChildren.Clear();
00041         }
00042     }
00043 }
```

7.123 SolitaireColumnChangeHandler.cs

```
00001 using UnityEngine;
00002
00003 namespace CardHouse.SampleGames.Solitaire
00004 {
00005     [RequireComponent(typeof(CardGroup))]
00006     public class SolitaireColumnChangeHandler : MonoBehaviour
00007     {
00008         CardGroup MyGroup;
00009
00010         void Awake()
00011         {
00012             MyGroup = GetComponent<CardGroup>();
00013         }
00014
00015         public void Refresh()
00016         {
00017             foreach (var card in MyGroup.MountedCards)
00018             {
00019                 if (card == MyGroup.Get())
00020                 {
00021                     card.SetFacing(CardFacing.FaceUp);
00022                 }
00023
00024                 card.GetComponent<Collider2D>().enabled = card.Facing == CardFacing.FaceUp;
00025             }
00026         }
00027     }
00028 }
```

7.124 SolitaireColumnDropGate.cs

```
00001 using System.Collections.Generic;
00002
00003 namespace CardHouse.SampleGames.Solitaire
00004 {
00005     public class SolitaireColumnDropGate : Gate<DropParams>
00006     {
00007         protected override bool IsUnlockedInternal(DropParams gateParams)
00008         {
00009             var topCard = gateParams.Target.Get();
00010             var pokerCard = gateParams.Card.GetComponent<PokerCard>();
00011             if (topCard == null)
00012             {
00013                 return pokerCard.Rank == 13; // King
00014             }
00015             else
00016             {
00017                 var topPokerCard = topCard.GetComponent<PokerCard>();
00018                 return !IsColorMatch(pokerCard, topPokerCard) && pokerCard.Rank == topPokerCard.Rank -
    1;
00019             }
00020         }
00021
00022         bool IsColorMatch(PokerCard a, PokerCard b)
00023         {
00024             var redSuits = new List<PokerSuit> { PokerSuit.Hearts, PokerSuit.Diamonds };
00025             return redSuits.Contains(a.Suit) == redSuits.Contains(b.Suit);
00026         }
00027     }
00028 }
```

7.125 SolitaireDeckClickHandler.cs

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

```
00002 using UnityEngine;
00003
00004 namespace CardHouse.SampleGames.Solitaire
00005 {
00006     [RequireComponent(typeof(CardGroup))]
00007     public class SolitaireDeckClickHandler : MonoBehaviour
00008     {
00009         public CardTransferOperator FlipHandler;
00010         public CardTransferOperator MoveToDeckHandler;
00011         public ShuffleOperator ShuffleHandler;
00012         public CardTransferOperator DealCardHandler;
00013         public List<TimedEvent> ResetEventChain;
00014
00015         CardGroup MyGroup;
00016
00017         void Awake()
00018         {
00019             MyGroup = GetComponent<CardGroup>();
00020         }
00021
00022         public void FlipOrReset()
00023         {
00024             if (MyGroup.MountedCards.Count == 0)
00025             {
00026                 StartCoroutine(TimedEvent.ExecuteChain(ResetEventChain));
00027             }
00028             else
00029             {
00030                 FlipHandler.Activate();
00031             }
00032         }
00033     }
00034 }
```

7.126 SolitaireScorePileDropGate.cs

```
00001 namespace CardHouse.SampleGames.Solitaire
00002 {
00003     public class SolitaireScorePileDropGate : Gate<DropParams>
00004     {
00005         protected override bool IsUnlockedInternal(DropParams gateParams)
00006         {
00007             var topCard = gateParams.Target.Get();
00008             var pokerCard = gateParams.Card.GetComponent<PokerCard>();
00009             if (topCard == null)
00010             {
00011                 return pokerCard.Rank == 1; // Ace
00012             }
00013             else
00014             {
00015                 var topPokerCard = topCard.GetComponent<PokerCard>();
00016                 return pokerCard.Suit == topPokerCard.Suit && pokerCard.Rank == topPokerCard.Rank + 1;
00017             }
00018         }
00019     }
00020 }
```

7.127 SolitaireSetup.cs

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 namespace CardHouse.SampleGames.Solitaire
00006 {
00007     public class SolitaireSetup : MonoBehaviour
00008     {
00009         public SeekerScriptable<Vector3> DealingStrategy;
00010         public CardGroup Deck;
00011         public List<CardGroup> Columns;
00012         public List<CardGroup> AllGroups;
00013         public EventChain ResetBoardEventChain;
00014
00015         bool CanDoSetup = true;
00016
00017         public void TryResetBoard()
00018         {
00019             if (CanDoSetup)
00020             {
```

```
00021                 ResetBoardEventChain.Activate();
00022             }
00023         }
00024
00025         public void DealCards()
00026         {
00027             StartCoroutine(RiffleDeal());
00028         }
00029
00030         IEnumerator RiffleDeal()
00031         {
00032             var cardCount = 0;
00033             for (var i = 0; i < Columns.Count; i++)
00034             {
00035                 cardCount += i + 1;
00036             }
00037
00038             var delayBetweenCards = 2f / (2f * cardCount);
00039             for (var i = 0; i < Columns.Count; i++)
00040             {
00041                 var column = Columns[i];
00042                 for (var j = 0; j < i; j++)
00043                 {
00044                     var card = Deck.Get();
00045                     column.Mount(card);
00046                     card.SetFacing(CardFacing.FaceDown);
00047
00048                     yield return new WaitForSeconds(delayBetweenCards);
00049                 }
00050                 var faceUpCard = Deck.Get();
00051                 column.Mount(faceUpCard, seekerSets: new SeekerSetList { new SeekerSet { Homing =
      DealingStrategy?.GetStrategy() } });
00052                 faceUpCard.SetFacing(CardFacing.FaceUp);
00053                 faceUpCard.GetComponent<Collider2D>().enabled = true;
00054
00055                 yield return new WaitForSeconds(delayBetweenCards);
00056             }
00057         }
00058
00059         public void PreventReset()
00060         {
00061             CanDoSetup = false;
00062         }
00063
00064         public void AllowReset()
00065         {
00066             CanDoSetup = true;
00067         }
00068     }
00069 }
```

7.128 SpreadManager.cs

```
00001 using System.Collections.Generic;
00002 using System.Linq;
00003 using TMPro;
00004 using UnityEngine;
00005 using UnityEngine.UI;
00006
00007 namespace CardHouse.SampleGames.Tarot
00008 {
00009     public class SpreadManager : MonoBehaviour
00010     {
00011         public Text SpreadLabel;
00012         public CardGroup Deck;
00013         public GameObject SpreadOrderLabelPrefab;
00014         public TMP_Text Key;
00015
00016         public List<TarotSpread> Spreads;
00017         List<GameObject> CurrentSpreadLabels = new List<GameObject>();
00018
00019         int CurrentSpreadIndex = 0;
00020
00021         void Start()
00022         {
00023             foreach (var spread in Spreads)
00024             {
00025                 foreach (var slot in spread.Slots)
00026                 {
00027                     slot.gameObject.SetActive(false);
00028                 }
00029             }
00030             AdjustSpread(0);
```

```
00031         }
00032
00033         public void NextSpread()
00034         {
00035             AdjustSpread(1);
00036         }
00037
00038         public void PreviousSpread()
00039         {
00040             AdjustSpread(-1);
00041         }
00042
00043         void AdjustSpread(int diff)
00044         {
00045             ShuffleCardsBackIn();
00046
00047             foreach (var label in CurrentSpreadLabels)
00048             {
00049                 Destroy(label);
00050             }
00051
00052             foreach (var slot in Spreads[CurrentSpreadIndex].Slots)
00053             {
00054                 slot.gameObject.SetActive(false);
00055             }
00056
00057             CurrentSpreadIndex = (CurrentSpreadIndex + diff) % Spreads.Count;
00058             while (CurrentSpreadIndex < 0)
00059             {
00060                 CurrentSpreadIndex += Spreads.Count;
00061             }
00062             SpreadLabel.text = Spreads[CurrentSpreadIndex].Name;
00063             Key.text = Spreads[CurrentSpreadIndex].Instructions;
00064
00065             CurrentSpreadLabels.Clear();
00066             for (var i = 0; i < Spreads[CurrentSpreadIndex].Slots.Count; i++)
00067             {
00068                 var slot = Spreads[CurrentSpreadIndex].Slots[i];
00069                 slot.gameObject.SetActive(true);
00070                 var label = Instantiate(SpreadOrderLabelPrefab, slot.transform);
00071                 label.GetComponent<TMP_Text>().text = (i + 1).ToString();
00072                 CurrentSpreadLabels.Add(label);
00073             }
00074         }
00075
00076         public void ShuffleCardsBackIn()
00077         {
00078             var areCardsInPlay = false;
00079             foreach (var slot in Spreads[CurrentSpreadIndex].Slots)
00080             {
00081                 foreach (var card in slot.MountedCards.ToList())
00082                 {
00083                     Deck.Mount(card);
00084                     areCardsInPlay = true;
00085                 }
00086             }
00087
00088             if (areCardsInPlay)
00089             {
00090                 Deck.Shuffle();
00091             }
00092         }
00093
00094         public void DealNextCard()
00095         {
00096             if (Deck.MountedCards.Count == 0)
00097                 return;
00098
00099             Spreads[CurrentSpreadIndex].FillNext(Deck.MountedCards[Deck.MountedCards.Count - 1]);
00100         }
00101     }
00102 }
```

7.129 TarotSpread.cs

```
00001 using System;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 namespace CardHouse.SampleGames.Tarot
00006 {
00007     [Serializable]
00008     public class TarotSpread
```

```
00009     {
00010         public string Name;
00011         [TextArea(1, 15)]
00012         public string Instructions;
00013         public List<CardGroup> Slots;
00014 
00015         public void FillNext(Card card)
00016         {
00017             foreach (var slot in Slots)
00018             {
00019                 if (slot.HasRoom())
00020                 {
00021                     var myAngle = UnityEngine.Random.Range(0f, 360f);
00022                     var tweak = Vector3.right * Mathf.Cos(myAngle) + Vector3.up * Mathf.Sin(myAngle) +
      Vector3.back;
00023                     var tweakCurve = AnimationCurve.EaseInOut(0, 0, 1, 0);
00024                     tweakCurve.AddKey(0.5f, 1f);
00025 
00026                     var cardSeeker = new TweakVector3Seeker(1f, AnimationCurve.EaseInOut(0, 0, 1, 1),
      UnityEngine.Random.Range(1f, 1.5f) * tweak, tweakCurve);
00027 
00028                     slot.Mount(card, seekerSets: new SeekerSetList { new SeekerSet { Card = card,
      Homing = cardSeeker } });
00029                     break;
00030                 }
00031             }
00032         }
00033     }
00034 }
```

7.130 WaypointTesterCard.cs

```
00001 using UnityEngine;
00002 
00003 namespace CardHouse.TestScenes
00004 {
00005     public class WaypointTesterCard : MonoBehaviour
00006     {
00007         public SeekerScriptableSet WaypointSeekers;
00008 
00009         public void Test()
00010         {
00011             var card = GetComponent<Card>();
00012             var tester = card?.Group.GetComponent<WaypointTesterGroup>();
00013             if (tester != null)
00014             {
00015                 tester.Test(card, WaypointSeekers.Homing, WaypointSeekers.Turning,
      WaypointSeekers.Scaling);
00016             }
00017         }
00018     }
00019 }
```

7.131 WaypointTesterGroup.cs

```
00001 using System.Collections.Generic;
00002 using System.Linq;
00003 using UnityEngine;
00004 
00005 namespace CardHouse.TestScenes
00006 {
00007     [RequireComponent(typeof(CardGroup))]
00008     public class WaypointTesterGroup : MonoBehaviour
00009     {
00010         public List<Transform> Waypoints;
00011 
00012         public void Test(Card card, SeekerScriptable<Vector3> homing, SeekerScriptable<float> turning,
      SeekerScriptable<float> scaling)
00013         {
00014             GetComponent<CardGroup>().Mount(card, seekerSets:
00015                 new SeekerSetList
00016                 {
00017                     new SeekerSet
00018                     {
00019                         Card = card,
00020                         Homing = homing.GetStrategy(Waypoints.Select(x => x.position).ToList()),
00021                         Turning = turning.GetStrategy(Waypoints.Select(x =>
      x.rotation.eulerAngles.z).ToList()),
00022                         Scaling = scaling.GetStrategy(Waypoints.Select(x => x.lossyScale.x).ToList())
```

```
00023                     }
00024                 });
00025         }
00026     }
00027 }
```

7.132 LaunchTutorialOption.cs

```
00001 #if UNITY_EDITOR
00002 using System.Collections.Generic;
00003 using System.Linq;
00004 using System.Reflection;
00005 using UnityEditor;
00006 using UnityEditor.SceneManagement;
00007 using UnityEngine;
00008
00009 namespace CardHouse.Tutorial
00010 {
00011     [InitializeOnLoad]
00012     public static class LaunchTutorialOption
00013     {
00014         static LaunchTutorialOption()
00015         {
00016             EditorApplication.playModeStateChanged += ReloadScenes;
00017         }
00018
00019         static void ReloadScenes(PlayModeStateChange change)
00020         {
00021             if (change != PlayModeStateChange.EnteredEditMode)
00022                 return;
00023
00024             var launchData = GetLaunchData();
00025             if (launchData == null)
00026                 return;
00027
00028             if (launchData.LaunchedTutorial)
00029             {
00030                 launchData.LaunchedTutorial = false;
00031                 EditorUtility.SetDirty(launchData);
00032                 AssetDatabase.SaveAssetIfDirty(launchData);
00033
00034                 bool first = true;
00035                 foreach (var scenePath in launchData.OpenScenes)
00036                 {
00037                     if (first)
00038                     {
00039                         EditorSceneManager.OpenScene(scenePath);
00040                         first = false;
00041                     }
00042                     else
00043                     {
00044                         EditorSceneManager.OpenScene(scenePath, OpenSceneMode.Additive);
00045                     }
00046                 }
00047
      EditorSceneManager.SetActiveScene(EditorSceneManager.GetSceneByPath(launchData.ActiveScene));
00048                 RemoveTutorialScenesFromBuildSettings();
00049             }
00050         }
00051
00052         [MenuItem("CardHouse/Launch Tutorial")]
00053         static void LaunchTutorial()
00054         {
00055             var launchData = GetLaunchData();
00056             if (launchData == null)
00057                 return;
00058
00059             launchData.LaunchedTutorial = true;
00060             launchData.OpenScenes = new List<string>();
00061             for (var i = 0; i < EditorSceneManager.sceneCount; i++)
00062             {
00063                 launchData.OpenScenes.Add(EditorSceneManager.GetSceneAt(i).path);
00064             }
00065             launchData.ActiveScene = EditorSceneManager.GetActiveScene().path;
00066             EditorUtility.SetDirty(launchData);
00067             AssetDatabase.SaveAssetIfDirty(launchData);
00068
00069             var requiredScenes = new List<string> {
      "Assets/CardHouse/Tutorial/Overlay/TutorialOverlay.unity" };
00070             var tutorials =
      AssetDatabase.LoadAssetAtPath<StringListScriptable>("Assets/CardHouse/Tutorial/TutorialSceneList.asset");
00071             foreach (var tutorialScene in tutorials.MyList)
00072             {
```

```
00073                 var sceneSubfolder = tutorialScene.Replace("(", "").Replace(")", " -");
00074
    requiredScenes.Add($"Assets/CardHouse/Tutorial/Lessons/{sceneSubfolder}/{tutorialScene}.unity");
00075             }
00076
00077             var buildScenes = EditorBuildSettings.scenes.ToList();
00078             foreach (var requiredScene in requiredScenes)
00079             {
00080                 if (!buildScenes.Any(x => x.path == requiredScene))
00081                 {
00082                     buildScenes.Add(new EditorBuildSettingsScene(requiredScene, true));
00083                 }
00084             }
00085
00086             EditorBuildSettings.scenes = buildScenes.ToArray();
00087
00088             EditorSceneManager.SaveCurrentModifiedScenesIfUserWantsTo();
00089             EditorSceneManager.OpenScene("Assets/CardHouse/Tutorial/Tutorial.unity");
00090
00091             var gameViewWindowType = typeof(Editor).Assembly.GetType("UnityEditor.GameView");
00092             var selectedSizeIndexProperty = gameViewWindowType.GetProperty("selectedSizeIndex",
00093                 BindingFlags.Instance | BindingFlags.Public | BindingFlags.NonPublic);
00094             var gameViewWindow = EditorWindow.GetWindow(gameViewWindowType);
00095             selectedSizeIndexProperty.SetValue(gameViewWindow, 1, null);
00096
00097             EditorApplication.isPlaying = true;
00098         }
00099
00100         static void RemoveTutorialScenesFromBuildSettings()
00101         {
00102             var scenes = new List<string> { "Assets/CardHouse/Tutorial/Overlay/TutorialOverlay.unity"
    };
00103             var tutorials =
    AssetDatabase.LoadAssetAtPath<StringListScriptable>("Assets/CardHouse/Tutorial/TutorialSceneList.asset");
00104             foreach (var tutorialScene in tutorials.MyList)
00105             {
00106                 var sceneSubfolder = tutorialScene.Replace("(", "").Replace(")", " -");
00107
    scenes.Add($"Assets/CardHouse/Tutorial/Lessons/{sceneSubfolder}/{tutorialScene}.unity");
00108             }
00109
00110             var buildScenes = EditorBuildSettings.scenes.ToList();
00111             foreach (var requiredScene in scenes)
00112             {
00113                 var result = buildScenes.FirstOrDefault(x => x.path == requiredScene);
00114                 if (result != null)
00115                 {
00116                     buildScenes.Remove(result);
00117                 }
00118             }
00119
00120             EditorBuildSettings.scenes = buildScenes.ToArray();
00121         }
00122
00123         static LaunchDataScriptable GetLaunchData()
00124         {
00125             return
    AssetDatabase.LoadAssetAtPath<LaunchDataScriptable>("Assets/CardHouse/Tutorial/LaunchData.asset");
00126         }
00127
00128         [MenuItem("CardHouse/Report a Bug")]
00129         static void OpenIssuesPage()
00130         {
00131             Application.OpenURL("https://github.com/pipeworks-studios/CardHouse/issues");
00132         }
00133     }
00134 }
00135 #endif
```

7.133 LaunchDataScriptable.cs

```
00001 using System.Collections.Generic;
00002 using UnityEngine;
00003
00004 namespace CardHouse.Tutorial
00005 {
00006     public class LaunchDataScriptable : ScriptableObject
00007     {
00008         public bool LaunchedTutorial;
00009         public string ActiveScene;
00010         public List<string> OpenScenes;
00011     }
00012 }
```

7.134 CardDragTutorial.cs

```
00001 using System.Collections;
00002 using TMPro;
00003 using UnityEngine;
00004 using UnityEngine.UI;
00005
00006 namespace CardHouse.Tutorial
00007 {
00008     public class CardDragTutorial : MonoBehaviour
00009     {
00010         public Slider DragSwellSlider;
00011         public TMP_Text DragSwellText;
00012         public Slider SeekerGainSlider;
00013         public TMP_Text SeekerGainText;
00014         public Toggle GrabOffsetToggle;
00015         public Slider XOffsetSlider;
00016         public TMP_Text XOffsetText;
00017         public Slider YOffsetSlider;
00018         public TMP_Text YOffsetText;
00019
00020         public Card Card;
00021
00022         bool HasInteractedWithSwellSlider;
00023
00024         private void Start()
00025         {
00026             ((ExponentialVector3SeekerScriptable)Dragging.Instance.DragHomingStrategy).XYGain = 12;
00027             GameObject.Find("SwellOutline").transform.localScale = Vector3.one *
      DragSwellSlider.value;
00028         }
00029
00030         public void AdjustDragSwellSlider()
00031         {
00032             HasInteractedWithSwellSlider = true;
00033             Card.GetComponent<DragOperator>().DragSwell = DragSwellSlider.value;
00034             DragSwellText.text = $"Drag Swell: x{DragSwellSlider.value:0.0}";
00035             GameObject.Find("SwellOutline").transform.localScale = Vector3.one *
      DragSwellSlider.value;
00036             GameObject.Find("SwellOutline").GetComponent<SpriteRenderer>().enabled = true;
00037             UpdateOffsetReticle();
00038         }
00039
00040         public void AdjustSeekerGainSlider()
00041         {
00042             ((ExponentialVector3SeekerScriptable)Dragging.Instance.DragHomingStrategy).XYGain =
      SeekerGainSlider.value;
00043             Dragging.Instance.UpdateStrategy();
00044             SeekerGainText.text = $"Seeker Gain: {SeekerGainSlider.value:0.0}";
00045         }
00046
00047         public void OnGrabOffsetToggled()
00048         {
00049             Dragging.Instance.SetNewOffsetOnGrab = GrabOffsetToggle.isOn;
00050             XOffsetSlider.interactable = !GrabOffsetToggle.isOn;
00051             YOffsetSlider.interactable = !GrabOffsetToggle.isOn;
00052             GameObject.Find("Reticle").GetComponent<SpriteRenderer>().enabled =
      !GrabOffsetToggle.isOn;
00053             GameObject.Find("SwellReticle").GetComponent<SpriteRenderer>().enabled =
      !GrabOffsetToggle.isOn;
00054             if (!GrabOffsetToggle.isOn)
00055             {
00056                 Dragging.Instance.GrabOffset.x = 0;
00057                 Dragging.Instance.GrabOffset.y = 0;
00058             }
00059         }
00060
00061         public void AdjustOffsetX()
00062         {
00063             Dragging.Instance.GrabOffset.x = XOffsetSlider.value;
00064             XOffsetText.text = $"X Offset: {XOffsetSlider.value:0.0}";
00065             UpdateOffsetReticle();
00066         }
00067
00068         public void AdjustOffsetY()
00069         {
00070             Dragging.Instance.GrabOffset.y = YOffsetSlider.value;
00071             XOffsetText.text = $"Y Offset: {YOffsetSlider.value:0.0}";
00072             UpdateOffsetReticle();
00073         }
00074
00075         void UpdateOffsetReticle()
00076         {
00077             GameObject.Find("Reticle").transform.localPosition = new Vector3(-XOffsetSlider.value,
      -YOffsetSlider.value, 0);
00078             GameObject.Find("SwellReticle").transform.localPosition = new Vector3(-1f /
      DragSwellSlider.value * XOffsetSlider.value, -1f / DragSwellSlider.value * YOffsetSlider.value, 0);
```

```
00079         }
00080
00081         public void ShowSwellOutline()
00082         {
00083             StartCoroutine(ShowSwellOutlineAfter(1f));
00084         }
00085
00086         IEnumerator ShowSwellOutlineAfter(float delay)
00087         {
00088             yield return new WaitForSeconds(delay);
00089             if (HasInteractedWithSwellSlider && Dragging.Instance.GetTarget() == null)
00090             {
00091                 GameObject.Find("SwellOutline").GetComponent<SpriteRenderer>().enabled = true;
00092             }
00093         }
00094     }
00095 }
```

7.135 GroupSetupTutorial.cs

```
00001 using System.Collections.Generic;
00002 using System.Linq;
00003 using TMPro;
00004 using UnityEngine;
00005 using UnityEngine.UI;
00006
00007 namespace CardHouse.Tutorial
00008 {
00009     public class GroupSetupTutorial : MonoBehaviour
00010     {
00011         public GroupSetup SetupComponent;
00012         public CardTransferOperator PullBackOperator;
00013         public CardGroup Deck;
00014
00015         public TMP_Text ASpadesText;
00016         public Slider ASpadesSlider;
00017         public TMP_Text QDiamondsText;
00018         public Slider QDiamondsSlider;
00019         public TMP_Text Hearts10Text;
00020         public Slider Hearts10Slider;
00021         public Toggle ShuffleToggle;
00022
00023         public void Setup()
00024         {
00025             PullBackOperator.Activate();
00026             foreach (var card in Deck.MountedCards.ToList())
00027             {
00028                 Deck.UnMount(card);
00029                 Destroy(card.gameObject);
00030             }
00031
00032             SetupComponent.DoSetup();
00033         }
00034
00035         public void AdjustShuffle()
00036         {
00037             SetupComponent.GroupsToShuffle = new List<CardGroup>();
00038             if (ShuffleToggle.isOn)
00039             {
00040                 SetupComponent.GroupsToShuffle.Add(Deck);
00041             }
00042         }
00043
00044         public void AdjustASpadesSlider()
00045         {
00046             ASpadesText.text = $"Ace of Spades: {ASpadesSlider.value:0}";
00047             var entry = SetupComponent.GroupPopulationList[0];
00048             entry.CardCount = Mathf.RoundToInt(ASpadesSlider.value);
00049             SetupComponent.GroupPopulationList[0] = entry;
00050         }
00051
00052         public void AdjustQDiamondsSlider()
00053         {
00054             QDiamondsText.text = $"Queen of Diamonds: {QDiamondsSlider.value:0}";
00055             var entry = SetupComponent.GroupPopulationList[1];
00056             entry.CardCount = Mathf.RoundToInt(QDiamondsSlider.value);
00057             SetupComponent.GroupPopulationList[1] = entry;
00058         }
00059
00060         public void AdjustHearts10Slider()
00061         {
00062             Hearts10Text.text = $"10 of Hearts: {Hearts10Slider.value:0}";
00063             var entry = SetupComponent.GroupPopulationList[2];
```

```
00064             entry.CardCount = Mathf.RoundToInt(Hearts10Slider.value);
00065             SetupComponent.GroupPopulationList[2] = entry;
00066         }
00067 
00068     }
00069 }
```

7.136 ClosestCardHighlighter.cs

```
00001 using UnityEngine;
00002 
00003 namespace CardHouse.Tutorial
00004 {
00005     public class ClosestCardHighlighter : MonoBehaviour
00006     {
00007         bool IsActive;
00008         CardGroup MyGroup;
00009 
00010         private void Start()
00011         {
00012             MyGroup = GetComponent<CardGroup>();
00013             CardGroup.OnNewActiveGroup += HandleNewActiveGroup;
00014         }
00015 
00016         void HandleNewActiveGroup(CardGroup group)
00017         {
00018             IsActive = group == MyGroup;
00019         }
00020 
00021         void Update()
00022         {
00023 
00024             var dragTarget = Dragging.Instance?.GetTarget();
00025             if (IsActive && dragTarget != null)
00026             {
00027 
00028                 var closestIndex = MyGroup.GetClosestMountedCardIndex(dragTarget.transform.position);
00029                 if (closestIndex == null)
00030                     return;
00031 
00032                 var diff = MyGroup.MountedCards[(int)closestIndex].transform.position -
      dragTarget.transform.position;
00033                 var insertPoint = diff.x > 0 ? closestIndex : closestIndex + 1;
00034 
00035                 for (var i = 0; i < MyGroup.MountedCards.Count; i++)
00036                 {
00037                     SetHighlightState(MyGroup.MountedCards[i], i == insertPoint);
00038                 }
00039             }
00040             else
00041             {
00042                 foreach (var card in MyGroup.MountedCards)
00043                 {
00044                     SetHighlightState(card, true);
00045                 }
00046             }
00047 
00048         }
00049 
00050         void SetHighlightState(Card card, bool state)
00051         {
00052             card.GetComponentInChildren<SpriteColorOperator>().Activate(state ? "Active" : "Dim");
00053         }
00054     }
00055 }
```

7.137 StackTutorial.cs

```
00001 using TMPro;
00002 using UnityEngine;
00003 using UnityEngine.UI;
00004 
00005 namespace CardHouse.Tutorial
00006 {
00007     public class StackTutorial : MonoBehaviour
00008     {
00009         public Slider XOffsetSlider;
00010         public TMP_Text XOffsetText;
00011         public Slider YOffsetSlider;
```

```
00012         public TMP_Text YOffsetText;
00013
00014         public StackLayout Stack;
00015
00016         public void AdjustXOffset()
00017         {
00018             SetXOffset(XOffsetSlider.value);
00019         }
00020
00021         public void AdjustYOffset()
00022         {
00023             SetYOffset(YOffsetSlider.value);
00024         }
00025
00026         void SetXOffset(float value)
00027         {
00028             XOffsetText.text = $"X Offset: {value:0.000}";
00029             Stack.MarginalCardOffset += Vector3.right * (value - Stack.MarginalCardOffset.x);
00030             XOffsetSlider.value = value;
00031             Stack.Apply(Stack.GetComponent<CardGroup>().MountedCards);
00032         }
00033         void SetYOffset(float value)
00034         {
00035             YOffsetText.text = $"Y Offset: {value:0.000}";
00036             Stack.MarginalCardOffset += Vector3.up * (value - Stack.MarginalCardOffset.y);
00037             YOffsetSlider.value = value;
00038             Stack.Apply(Stack.GetComponent<CardGroup>().MountedCards);
00039         }
00040
00041         public void UseColumnPreset()
00042         {
00043             SetXOffset(0);
00044             SetYOffset(-0.2f);
00045         }
00046
00047         public void UseDeckPreset()
00048         {
00049             SetXOffset(0.03f);
00050             SetYOffset(0.03f);
00051         }
00052
00053         public void UseCompactDeckPreset()
00054         {
00055             SetXOffset(0.003f);
00056             SetYOffset(0.003f);
00057         }
00058
00059         public void UseRowPreset()
00060         {
00061             SetXOffset(1f);
00062             SetYOffset(0f);
00063         }
00064     }
00065 }
```

7.138 SplayTutorial.cs

```
00001 using TMPro;
00002 using UnityEngine;
00003 using UnityEngine.UI;
00004
00005 namespace CardHouse.Tutorial
00006 {
00007     public class SplayTutorial : MonoBehaviour
00008     {
00009         public Slider XScaleSlider;
00010         public TMP_Text XScaleText;
00011         public Slider ArcMarginSlider;
00012         public TMP_Text ArcMarginText;
00013         public Slider XOffsetSlider;
00014         public TMP_Text XOffsetText;
00015         public Slider YOffsetSlider;
00016         public TMP_Text YOffsetText;
00017
00018         public CardGroup Deck;
00019         public SplayLayout Splay;
00020         public SpriteRenderer Reticle;
00021         CardGroup Group;
00022
00023         bool HasAdjustedOffset;
00024
00025         void Start()
00026         {
```

```
00027             Group = Splay.GetComponent<CardGroup>();
00028         }
00029
00030         public void AdjustXScale()
00031         {
00032             XScaleText.text = $"X Scale: {XScaleSlider.value:0.0}";
00033             Splay.transform.localScale += Vector3.right * (XScaleSlider.value -
      Splay.transform.localScale.x);
00034             Splay.Apply(Group.MountedCards);
00035         }
00036
00037         public void AdjustArcMargin()
00038         {
00039             ArcMarginText.text = $"Arc Margin: {ArcMarginSlider.value:0.0}";
00040             Splay.ArcMargin = ArcMarginSlider.value;
00041             Splay.Apply(Group.MountedCards);
00042         }
00043
00044         public void AdjustXOffset()
00045         {
00046             Reticle.enabled = true;
00047             XOffsetText.text = $"X Offset: {XOffsetSlider.value:0.0}";
00048             Splay.ArcCenterOffset += Vector2.right * (XOffsetSlider.value - Splay.ArcCenterOffset.x);
00049             Reticle.transform.position = Splay.transform.position + (Vector3)Splay.ArcCenterOffset;
00050             Splay.Apply(Group.MountedCards);
00051         }
00052
00053         public void AdjustYOffset()
00054         {
00055             Reticle.enabled = true;
00056             YOffsetText.text = $"Y Offset: {YOffsetSlider.value:0.0}";
00057             Splay.ArcCenterOffset += Vector2.up * (YOffsetSlider.value - Splay.ArcCenterOffset.y);
00058             Reticle.transform.position = Splay.transform.position + (Vector3)Splay.ArcCenterOffset;
00059             Splay.Apply(Group.MountedCards);
00060         }
00061     }
00062 }
```

7.139 GridTutorial.cs

```
00001 using TMPro;
00002 using UnityEngine;
00003 using UnityEngine.UI;
00004
00005 namespace CardHouse.Tutorial
00006 {
00007     public class GridTutorial : MonoBehaviour
00008     {
00009         public Slider CardsPerRowSlider;
00010         public TMP_Text CardsPerRowText;
00011         public Slider CardLimitSlider;
00012         public TMP_Text CardLimitText;
00013         public Slider XScaleSlider;
00014         public TMP_Text XScaleText;
00015         public Slider YScaleSlider;
00016         public TMP_Text YScaleText;
00017
00018         public CardGroup Deck;
00019         public CardGridLayout Grid;
00020         CardGroup Group;
00021
00022         void Start()
00023         {
00024             Group = Grid.GetComponent<CardGroup>();
00025         }
00026
00027         public void AdjustCardsPerRow()
00028         {
00029             CardsPerRowText.text = $"Cards Per Row: {CardsPerRowSlider.value:0}";
00030             Grid.CardsPerRow = Mathf.RoundToInt(CardsPerRowSlider.value);
00031             Grid.Apply(Group.MountedCards);
00032         }
00033
00034         public void AdjustCardLimit()
00035         {
00036             CardLimitText.text = $"Card Limit: {CardLimitSlider.value:0}";
00037             Grid.CardLimit = Mathf.RoundToInt(CardLimitSlider.value);
00038             while (Group.MountedCards.Count > Grid.CardLimit)
00039             {
00040                 Deck.Mount(Group.Get());
00041             }
00042             Grid.Apply(Group.MountedCards);
00043         }
```

```
00044
00045         public void AdjustXScale()
00046         {
00047             XScaleText.text = $"X Scale: {XScaleSlider.value:0.0}";
00048             Grid.transform.localScale += Vector3.right * (XScaleSlider.value -
      Grid.transform.localScale.x);
00049             Grid.Apply(Group.MountedCards);
00050         }
00051
00052         public void AdjustYScale()
00053         {
00054             YScaleText.text = $"Y Scale: {YScaleSlider.value:0.0}";
00055             Grid.transform.localScale += Vector3.up * (YScaleSlider.value -
      Grid.transform.localScale.y);
00056             Grid.Apply(Group.MountedCards);
00057         }
00058
00059     }
00060 }
```

7.140 DiscardAllCardsOperator.cs

```
00001 using System.Linq;
00002 using UnityEngine;
00003
00004 namespace CardHouse.Tutorial
00005 {
00006     public class DiscardAllCardsOperator : MonoBehaviour
00007     {
00008         public SeekerScriptableSet DiscardSeekers;
00009         public SeekerScriptableSet TargetDiscardSeekers;
00010
00011         public void Activate()
00012         {
00013             var boardGroups = GroupRegistry.Instance?.Groups.Where(x => x.Name ==
      GroupName.Board).Select(x => x.Group);
00014             if (boardGroups != null)
00015             {
00016                 var seekerSets = new SeekerSetList();
00017
00018                 var presentationTransform =
      PhaseManager.Instance?.CurrentPhase?.CardPresentationPosition;
00019                 if (presentationTransform != null)
00020                 {
00021                     seekerSets.Add(new SeekerSet
00022                     {
00023                         Card = GetComponent<Card>(),
00024                         Homing = DiscardSeekers.Homing.GetStrategy(presentationTransform.position),
00025                         Turning =
      DiscardSeekers.Turning.GetStrategy(CardHouse.Utils.CorrectAngle(presentationTransform.rotation.eulerAngles.z)),
00026                         Scaling =
      DiscardSeekers.Scaling.GetStrategy(presentationTransform.lossyScale.x)
00027                     });
00028                 }
00029
00030                 foreach (var boardGroup in boardGroups)
00031                 {
00032                     foreach (var target in boardGroup.MountedCards.ToArray())
00033                     {
00034                         seekerSets.Add(new SeekerSet { Card = target, Homing =
      TargetDiscardSeekers.Homing?.GetStrategy() });
00035                     }
00036
00037                     foreach (var target in boardGroup.MountedCards.ToArray())
00038                     {
00039                         target.GetDiscardGroup()?.Mount(target,
00040                             seekerSets: seekerSets,
00041                             seekersForUnmounting: new SeekerSet { Homing =
      DiscardSeekers.Homing?.GetStrategy() }
00042                         );
00043                     }
00044                 }
00045             }
00046         }
00047     }
00048 }
```

7.141 SpriteOperatorTutorial.cs

```
00001 using UnityEngine;
```

```
00002
00003 namespace CardHouse.Tutorial
00004 {
00005     public class SpriteOperatorTutorial : MonoBehaviour
00006     {
00007         public MultiSpriteOperator ColorOperator;
00008         public SpriteImageOperator ImageOperator;
00009
00010         public static SpriteOperatorTutorial Instance;
00011
00012
00013         private void Awake()
00014         {
00015             Instance = this;
00016         }
00017
00018         public void RegisterColorVote(Object voter, string vote)
00019         {
00020             ColorOperator.Activate(vote, voter);
00021         }
00022
00023         public void RemoveColorVote(Object voter)
00024         {
00025             ColorOperator.Remove(voter);
00026         }
00027
00028         public void RegisterImageVote(Object voter, string vote)
00029         {
00030             ImageOperator.Activate(vote, voter);
00031         }
00032
00033         public void RemoveImageVote(Object voter)
00034         {
00035             ImageOperator.Remove(voter);
00036         }
00037     }
00038 }
```

7.142 SpriteVoterTutorial.cs

```
00001 using TMPro;
00002 using UnityEngine;
00003 using UnityEngine.Events;
00004
00005 namespace CardHouse.Tutorial
00006 {
00007     public class SpriteVoterTutorial : MonoBehaviour
00008     {
00009         public UnityEvent OnStart;
00010
00011         private void Start()
00012         {
00013             OnStart?.Invoke();
00014         }
00015
00016         public void OnColorDropdownUpdated()
00017         {
00018             var value = GetComponent<TMP_Dropdown>().value;
00019             switch (value)
00020             {
00021                 case 0:
00022                     RemoveColorVote();
00023                     break;
00024                 case 1:
00025                     RegisterColorVote("Active");
00026                     break;
00027                 case 2:
00028                     RegisterColorVote("Dim");
00029                     break;
00030             }
00031         }
00032
00033         public void OnImageDropdownUpdated()
00034         {
00035             var value = GetComponent<TMP_Dropdown>().value;
00036             switch (value)
00037             {
00038                 case 0:
00039                     RemoveImageVote();
00040                     break;
00041                 case 1:
00042                     RegisterImageVote("Bat");
00043                     break;
```

```
00044                 case 2:
00045                     RegisterImageVote("Spider");
00046                     break;
00047             }
00048         }
00049 
00050         void RegisterColorVote(string vote)
00051         {
00052             SpriteOperatorTutorial.Instance.RegisterColorVote(this, vote);
00053         }
00054 
00055         void RemoveColorVote()
00056         {
00057             SpriteOperatorTutorial.Instance.RemoveColorVote(this);
00058         }
00059 
00060         void RegisterImageVote(string vote)
00061         {
00062             SpriteOperatorTutorial.Instance.RegisterImageVote(this, vote);
00063         }
00064 
00065         void RemoveImageVote()
00066         {
00067             SpriteOperatorTutorial.Instance.RemoveImageVote(this);
00068         }
00069     }
00070 }
```

7.143 TransferOperatorTutorialUI.cs

```
00001 using TMPro;
00002 using UnityEngine;
00003 using UnityEngine.UI;
00004 
00005 namespace CardHouse.Tutorial
00006 {
00007     public class TransferOperatorTutorialUI : MonoBehaviour
00008     {
00009         public TMP_Dropdown GrabFromDropdown;
00010         public TMP_Dropdown SendToDropdown;
00011         public TMP_Text NumberToTransferText;
00012         public Slider NumberToTransferSlider;
00013         public TMP_Text FlipSpeedText;
00014         public Slider FlipSpeedSlider;
00015         public CardTransferOperator Operator;
00016 
00017         public void AdjustNumberToTransfer()
00018         {
00019             NumberToTransferText.text = $"# to Transfer: {NumberToTransferSlider.value:0}";
00020             Operator.NumberToTransfer = Mathf.RoundToInt(NumberToTransferSlider.value);
00021         }
00022 
00023         public void AdjustFlipSpeed()
00024         {
00025             FlipSpeedText.text = $"Flip Speed: {FlipSpeedSlider.value:0.00}";
00026             Operator.FlipSpeed = FlipSpeedSlider.value;
00027         }
00028 
00029         public void AdjustGrabFrom()
00030         {
00031             var i = GrabFromDropdown.value;
00032             switch (i)
00033             {
00034                 case 0:
00035                     Operator.GrabFrom = GroupTargetType.Last;
00036                     break;
00037                 case 1:
00038                     Operator.GrabFrom = GroupTargetType.First;
00039                     break;
00040                 case 2:
00041                     Operator.GrabFrom = GroupTargetType.Random;
00042                     break;
00043             }
00044         }
00045 
00046         public void AdjustSendTo()
00047         {
00048             var i = SendToDropdown.value;
00049             switch (i)
00050             {
00051                 case 0:
00052                     Operator.SendTo = GroupTargetType.Last;
00053                     break;
```

```
00054                 case 1:
00055                     Operator.SendTo = GroupTargetType.First;
00056                     break;
00057                 case 2:
00058                     Operator.SendTo = GroupTargetType.Random;
00059                     break;
00060             }
00061         }
00062     }
00063 }
```

7.144 SeekerTutorial.cs

```
00001 using System;
00002 using System.Collections.Generic;
00003 using TMPro;
00004 using UnityEngine;
00005 
00006 namespace CardHouse.Tutorial
00007 {
00008     public class SeekerTutorial : MonoBehaviour
00009     {
00010         public TMP_Dropdown HomingDropdown;
00011         public TMP_Dropdown TurningDropdown;
00012         public TMP_Dropdown ScalingDropdown;
00013 
00014         public List<CardGroup> Stacks;
00015 
00016         public List<StringSeekerKVP> SeekerKVPs;
00017 
00018         public Transform Waypoint;
00019 
00020         public void Transfer(int i)
00021         {
00022             var card = Stacks[0].Get();
00023             if (card == null)
00024                 return;
00025 
00026             Seeker<Vector3> homing = null;
00027             var homingKey = HomingDropdown.options[HomingDropdown.value].text;
00028             foreach (var kvp in SeekerKVPs)
00029             {
00030                 if (kvp.Key == homingKey)
00031                 {
00032                     homing =
      ((SeekerScriptable<Vector3>)kvp.Value).GetStrategy(kvp.Key.Contains("Vector3") ? Waypoint.position :
      null);
00033                     break;
00034                 }
00035             }
00036             Seeker<float> turning = null;
00037             var turningKey = TurningDropdown.options[TurningDropdown.value].text;
00038             foreach (var kvp in SeekerKVPs)
00039             {
00040                 if (kvp.Key == turningKey)
00041                 {
00042                     turning =
      ((SeekerScriptable<float>)kvp.Value).GetStrategy(kvp.Key.Contains("Angle") ? Waypoint.eulerAngles.z :
      null);
00043                     break;
00044                 }
00045             }
00046             Seeker<float> scaling = null;
00047             var scalingKey = ScalingDropdown.options[ScalingDropdown.value].text;
00048             foreach (var kvp in SeekerKVPs)
00049             {
00050                 if (kvp.Key == scalingKey)
00051                 {
00052                     scaling =
      ((SeekerScriptable<float>)kvp.Value).GetStrategy(kvp.Key.Contains("Float") ? Waypoint.lossyScale.y :
      null);
00053                     break;
00054                 }
00055             }
00056 
00057             Stacks[i].Mount(card, seekerSets: new SeekerSetList { new SeekerSet { Card = card, Homing
      = homing, Scaling = scaling, Turning = turning } });
00058         }
00059     }
00060 
00061     [Serializable]
00062     public class StringSeekerKVP
00063     {
```

```
00064         public string Key;
00065         public ScriptableObject Value;
00066     }
00067 }
```

7.145 EventChainsTutorial.cs

```
00001 using TMPro;
00002 using UnityEngine;
00003
00004 namespace CardHouse.Tutorial
00005 {
00006     public class EventChainsTutorial : MonoBehaviour
00007     {
00008         public EventChain NoChaining;
00009         public EventChain Chaining;
00010         public EventChain SafeChaining;
00011
00012         public TMP_Dropdown Dropdown;
00013
00014         public void StartTransition()
00015         {
00016             switch (Dropdown.value)
00017             {
00018                 case 0:
00019                     NoChaining.Activate();
00020                     break;
00021                 case 1:
00022                     Chaining.Activate();
00023                     break;
00024                 case 2:
00025                     SafeChaining.Activate();
00026                     break;
00027             }
00028         }
00029     }
00030 }
```

7.146 ValidDragTutorial.cs

```
00001 using TMPro;
00002 using UnityEngine;
00003
00004 namespace CardHouse.Tutorial
00005 {
00006     public class ValidDragTutorial : MonoBehaviour
00007     {
00008         public PhaseManager PhaseManager;
00009
00010         public CardGroup GroupA;
00011         public CardGroup GroupB;
00012         public CardGroup GroupC;
00013         public CardGroup GroupD;
00014
00015         public TMP_Dropdown Dropdown10;
00016         public TMP_Dropdown Dropdown01;
00017         public TMP_Dropdown Dropdown11;
00018         public TMP_Dropdown Dropdown02;
00019         public TMP_Dropdown Dropdown12;
00020
00021         public void UpdateDropdown10()
00022         {
00023             UpdateDropdown(0, false, Dropdown10.value);
00024         }
00025
00026         public void UpdateDropdown01()
00027         {
00028             UpdateDropdown(1, true, Dropdown01.value);
00029         }
00030         public void UpdateDropdown11()
00031         {
00032             UpdateDropdown(1, false, Dropdown11.value);
00033         }
00034         public void UpdateDropdown02()
00035         {
00036             UpdateDropdown(2, true, Dropdown02.value);
00037         }
00038         public void UpdateDropdown12()
00039         {
```

```
00040             UpdateDropdown(2, false, Dropdown12.value);
00041         }
00042
00043         void UpdateDropdown(int element, bool isSource, int groupIndex)
00044         {
00045             var drag = PhaseManager.Phases[0].ValidDrags[element];
00046             if (isSource)
00047             {
00048                 drag.Source = GetGroup(groupIndex);
00049             }
00050             else
00051             {
00052                 drag.Destination = GetGroup(groupIndex);
00053             }
00054         }
00055
00056         CardGroup GetGroup(int i)
00057         {
00058             switch (i)
00059             {
00060                 case 0:
00061                     return GroupA;
00062                 case 1:
00063                     return GroupB;
00064                 case 2:
00065                     return GroupC;
00066                 case 3:
00067                     return GroupD;
00068             }
00069             return GroupA;
00070         }
00071     }
00072 }
```

7.147 MatureCropDragGate.cs

```
00001 namespace CardHouse.Tutorial
00002 {
00003     public class MatureCropDragGate : Gate<NoParams>
00004     {
00005         Card MyCard;
00006
00007         private void Awake()
00008         {
00009             MyCard = GetComponent<Card>();
00010         }
00011
00012         protected override bool IsUnlockedInternal(NoParams gateParams)
00013         {
00014             if (MyCard.Group != GroupRegistry.Instance.Get(GroupName.Board, null))
00015                 return true;
00016
00017             return MyCard.GetComponent<Plant>()?.CanBeWatered() != true;
00018         }
00019     }
00020 }
```

7.148 PhaseLabelUpdater.cs

```
00001 using TMPro;
00002 using UnityEngine;
00003
00004 namespace CardHouse.Tutorial
00005 {
00006     public class PhaseLabelUpdater : MonoBehaviour
00007     {
00008         public TMP_Text PhaseText;
00009
00010         public void UpdatePhaseLabel()
00011         {
00012             PhaseText.text = PhaseManager.Instance.CurrentPhase.Name;
00013         }
00014     }
00015 }
```

7.149 Plant.cs

```
00001 using System.Collections.Generic;
00002 using TMPro;
00003 using UnityEngine;
00004
00005 namespace CardHouse.Tutorial
00006 {
00007     public class Plant : MonoBehaviour
00008     {
00009         public TMP_Text NameText;
00010         public TMP_Text DescriptionText;
00011         public SpriteRenderer Sprite;
00012         public GameObject CostJewel;
00013         public TMP_Text CostText;
00014
00015         public List<PlantGrowthScriptable> PossiblePlants;
00016         List<PlantMaturityInfo> Stages;
00017
00018         public int Value = 10;
00019
00020         int WaterLevel = -1;
00021
00022         private void Start()
00023         {
00024             Stages = PossiblePlants[UnityEngine.Random.Range(0, PossiblePlants.Count)].Stages;
00025             Water();
00026         }
00027
00028         public void Water()
00029         {
00030             if (CanBeWatered())
00031             {
00032                 WaterLevel++;
00033                 NameText.text = Stages[WaterLevel].Name;
00034                 DescriptionText.text = Stages[WaterLevel].Description;
00035                 Sprite.sprite = Stages[WaterLevel].Sprite;
00036
00037                 if (!CanBeWatered() && CostJewel != null)
00038                 {
00039                     CostJewel.SetActive(true);
00040                     CostText.text = Value.ToString();
00041                 }
00042             }
00043         }
00044
00045         public void HideCost()
00046         {
00047             CostJewel.SetActive(false);
00048         }
00049
00050         public void Payoff()
00051         {
00052             CurrencyRegistry.Instance.AdjustCurrency("Gold", PhaseManager.Instance.PlayerIndex,
      Value);
00053         }
00054
00055         public bool CanBeWatered()
00056         {
00057             return WaterLevel < Stages.Count - 1;
00058         }
00059     }
00060 }
```

7.150 PlantGrowthScriptable.cs

```
00001 using System;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 namespace CardHouse.Tutorial
00006 {
00007     public class PlantGrowthScriptable : ScriptableObject
00008     {
00009         public List<PlantMaturityInfo> Stages;
00010     }
00011
00012     [Serializable]
00013     public class PlantMaturityInfo
00014     {
00015         public string Name;
00016         public string Description;
00017         public Sprite Sprite;
```

```
00018     }
00019 }
```

7.151 WaterPlantAction.cs

```
00001 namespace CardHouse.Tutorial
00002 {
00003     public class WaterPlantAction : CardTargetCardOperator
00004     {
00005         protected override void ActOnTarget()
00006         {
00007             var plant = Target.GetComponent<Plant>();
00008             if (plant != null)
00009             {
00010                 plant.Water();
00011             }
00012         }
00013     }
00014 }
```

7.152 WaterTargetPlantGate.cs

```
00001 namespace CardHouse.Tutorial
00002 {
00003     public class WaterTargetPlantGate : Gate<TargetCardParams>
00004     {
00005         protected override bool IsUnlockedInternal(TargetCardParams gateParams)
00006         {
00007             return gateParams.Target.GetComponent<Plant>()?.CanBeWatered() ?? false;
00008         }
00009     }
00010 }
```

7.153 PresentationPointTutorial.cs

```
00001 using System.Collections.Generic;
00002 using UnityEngine;
00003
00004 namespace CardHouse.Tutorial
00005 {
00006     public class PresentationPointTutorial : MonoBehaviour
00007     {
00008         public int PlayerIndex;
00009
00010         public Turning ParentTurning;
00011         public Scaling ParentScaling;
00012         public Turning ButtonParentTurning;
00013         public Scaling ButtonParentScaling;
00014
00015         public float RotationMin;
00016         public float RotationMax;
00017         public int RotationSteps;
00018
00019         public float ScaleMin;
00020         public float ScaleMax;
00021         public int ScaleSteps;
00022
00023         List<float> Rotations = new List<float>();
00024         int RotationI;
00025         List<float> Scales = new List<float>();
00026         int ScaleI;
00027
00028         private void Start()
00029         {
00030             for (var r = RotationMin; r <= RotationMax; r += (RotationMax - RotationMin) /
      RotationSteps)
00031             {
00032                 Rotations.Add(r);
00033             }
00034             RotationI = Mathf.CeilToInt(RotationSteps / 2f);
00035
00036             for (var s = ScaleMin; s <= ScaleMax; s += (ScaleMax - ScaleMin) / ScaleSteps)
00037             {
00038                 Scales.Add(s);
00039             }
```

```
00040             ScaleI = Scales.IndexOf(1f);
00041         }
00042
00043         public void Rotate(int shift)
00044         {
00045             RotationI = Mathf.Clamp(RotationI + shift, 0, Rotations.Count - 1);
00046             ParentTurning.StartSeeking(Rotations[RotationI], useLocalSpace: true);
00047             ButtonParentTurning.StartSeeking(-Rotations[RotationI], useLocalSpace: true);
00048         }
00049
00050         public void Scale(int shift)
00051         {
00052             ScaleI = Mathf.Clamp(ScaleI + shift, 0, Scales.Count - 1);
00053             ParentScaling.StartSeeking(Scales[ScaleI]);
00054             ButtonParentScaling.StartSeeking(1f / Scales[ScaleI], useLocalSpace: true);
00055         }
00056
00057         public void UpdateCameraPosition()
00058         {
00059             if (PlayerIndex != PhaseManager.Instance?.CurrentPhase.PlayerIndex)
00060                 return;
00061
00062             PhaseManager.Instance?.SetCameraPosition(transform);
00063         }
00064     }
00065 }
```

7.154 MultiBoardTutorial.cs

```
00001 using System;
00002 using System.Collections.Generic;
00003 using TMPro;
00004 using UnityEngine;
00005 using UnityEngine.UI;
00006
00007 namespace CardHouse.Tutorial
00008 {
00009     public class MultiBoardTutorial : MonoBehaviour
00010     {
00011         [Serializable]
00012         public class InstructionImagePair
00013         {
00014             public Sprite Image;
00015             [TextArea]
00016             public string Text;
00017         }
00018
00019         public MultiplayerBoardSetup SetupScript;
00020         public TMP_Text PlayerCountLabel;
00021         public Slider PlayerCountSlider;
00022         public TMP_Text SpacingLabel;
00023         public Slider SpacingSlider;
00024         public Button SetupButton;
00025
00026         public GameObject InstructionsRoot;
00027         public Image InstructionsImage;
00028         public TMP_Text InstructionsText;
00029         public List<InstructionImagePair> Instructions;
00030         public TMP_Text PageNumberText;
00031         public Button ForwardButton;
00032         public Button BackButton;
00033         public int InstructionIndex;
00034
00035         public GameObject CommonArea;
00036
00037
00038         private void Start()
00039         {
00040             SandboxManager.MultiBoardTutorial = this;
00041             UpdateInstructions();
00042         }
00043
00044         private void OnDestroy()
00045         {
00046             SandboxManager.MultiBoardTutorial = null;
00047         }
00048
00049         void UpdateInstructions()
00050         {
00051             InstructionsImage.sprite = Instructions[InstructionIndex].Image;
00052             InstructionsText.text = Instructions[InstructionIndex].Text;
00053             PageNumberText.text = $"{InstructionIndex + 1} / {Instructions.Count}";
00054             BackButton.gameObject.SetActive(InstructionIndex > 0);
```

```
00055             ForwardButton.gameObject.SetActive(InstructionIndex < Instructions.Count - 1);
00056
00057             CommonArea.SetActive(InstructionIndex > 4);
00058
00059             switch (InstructionIndex)
00060             {
00061                 case 0:
00062                     foreach (var board in SetupScript.GetAllBoards())
00063                     {
00064                         board.transform.GetChild(0).gameObject.SetActive(true);
00065                     }
00066                     PlayerCountLabel.color = Color.yellow;
00067                     SetSelectableColor(PlayerCountSlider, Color.yellow);
00068                     SetSelectableColor(SetupButton, Color.yellow);
00069                     break;
00070                 case 1:
00071                     SpacingLabel.color = Color.yellow;
00072                     SetSelectableColor(SpacingSlider, Color.yellow);
00073                     SetSelectableColor(SetupButton, Color.yellow);
00074                     break;
00075             }
00076         }
00077
00078         void SetSelectableColor(Selectable button, Color color)
00079         {
00080             var buttonColors = button.colors;
00081             buttonColors.normalColor = color;
00082             buttonColors.selectedColor = color;
00083             button.colors = buttonColors;
00084         }
00085
00086         void TearDownInstructions()
00087         {
00088             switch (InstructionIndex)
00089             {
00090                 case 0:
00091                     foreach (var board in SetupScript.GetAllBoards())
00092                     {
00093                         board.transform.GetChild(0).gameObject.SetActive(false);
00094                     }
00095                     PlayerCountLabel.color = Color.white;
00096                     SetSelectableColor(PlayerCountSlider, Color.white);
00097                     SetSelectableColor(SetupButton, Color.white);
00098
00099                     break;
00100                 case 1:
00101                     SpacingLabel.color = Color.white;
00102                     SetSelectableColor(SpacingSlider, Color.white);
00103                     SetSelectableColor(SetupButton, Color.white);
00104                     break;
00105             }
00106         }
00107
00108         public void InstructionsForward()
00109         {
00110             TearDownInstructions();
00111             InstructionIndex++;
00112             if (InstructionIndex >= Instructions.Count)
00113             {
00114                 InstructionsRoot.SetActive(false);
00115             }
00116             else
00117             {
00118                 UpdateInstructions();
00119             }
00120         }
00121
00122         public void InstructionsBackward()
00123         {
00124             InstructionIndex = Mathf.Max(0, InstructionIndex - 1);
00125
00126             UpdateInstructions();
00127         }
00128
00129         public void SetupBoard()
00130         {
00131             SetupScript.PlayerCount = Mathf.RoundToInt(PlayerCountSlider.value);
00132             SetupScript.SpacingMultiplier = SpacingSlider.value;
00133
00134             SetupScript.Setup(InstructionIndex > 1);
00135         }
00136
00137         public void UpdatePlayerCount()
00138         {
00139             PlayerCountLabel.text = $"Player Count: {PlayerCountSlider.value: 0}";
00140         }
00141
```

```
00142         public void UpdateSpacingMultiplier()
00143         {
00144             SpacingLabel.text = $"Spacing Multiplier: {SpacingSlider.value: 0.00}";
00145         }
00146     }
00147 }
```

7.155 OutLinks.cs

```
00001 using TMPro;
00002 using UnityEngine;
00003 using UnityEngine.EventSystems;
00004
00005 namespace CardHouse.Tutorial
00006 {
00007     public class OutLinks : MonoBehaviour, IPointerClickHandler
00008     {
00009         public TMP_Text Text;
00010
00011         public void OnPointerClick(PointerEventData eventData)
00012         {
00013             var linkIndex = TMP_TextUtilities.FindIntersectingLink(Text, Input.mousePosition, null);
00014             var linkId = Text.textInfo.linkInfo[linkIndex].GetLinkID();
00015
00016             var url = linkId switch
00017             {
00018                 "GitHubIssues" => "https://github.com/pipeworks-studios/CardHouse/issues",
00019                 "Pipeworks" => "https://www.pipeworks.com/",
00020                 _ => ""
00021             };
00022
00023             if (url != "")
00024             {
00025                 Application.OpenURL(url);
00026             }
00027         }
00028     }
00029 }
```

7.156 SandboxManager.cs

```
00001 using TMPro;
00002 using UnityEngine;
00003 using UnityEngine.SceneManagement;
00004
00005 namespace CardHouse.Tutorial
00006 {
00007     public class SandboxManager : MonoBehaviour
00008     {
00009         public GameObject TutorialButtonPrefab;
00010         public Transform TutorialListRoot;
00011         public StringListScriptable Tutorials;
00012         int currentTutorial = -1;
00013         public Animator SidebarAnimator;
00014         public TMP_Text TitleText;
00015         public GameObject NextButton;
00016         public GameObject PreviousButton;
00017         public GameObject ResetButton;
00018
00019         public static MultiBoardTutorial MultiBoardTutorial;
00020
00021         public void Start()
00022         {
00023             for (var i = 0; i < TutorialListRoot.childCount; i++)
00024             {
00025                 Destroy(TutorialListRoot.GetChild(i).gameObject);
00026             }
00027             for (var i = Tutorials.MyList.Count - 1; i >= 0; i--)
00028             {
00029                 var newButton = Instantiate(TutorialButtonPrefab);
00030                 newButton.GetComponent<TutorialButton>().Setup(Tutorials.MyList[i], this);
00031                 newButton.transform.SetParent(TutorialListRoot.transform, false);
00032             }
00033
00034             TitleText.text = "";
00035             PreviousButton.SetActive(false);
00036             ResetButton.SetActive(false);
00037         }
00038
```

```
00039         public void Reset()
00040         {
00041             SetupCurrentTutorial();
00042         }
00043 
00044         public void GoToNext()
00045         {
00046             if (MultiBoardTutorial != null && MultiBoardTutorial.InstructionIndex <
      MultiBoardTutorial.Instructions.Count - 1)
00047             {
00048                 MultiBoardTutorial.InstructionsForward();
00049             }
00050             else
00051             {
00052                 if (currentTutorial < Tutorials.MyList.Count - 1)
00053                 {
00054                     currentTutorial++;
00055                     SetupCurrentTutorial();
00056                 }
00057             }
00058         }
00059 
00060         public void GoToPrevious()
00061         {
00062             if (MultiBoardTutorial != null && MultiBoardTutorial.InstructionIndex > 0)
00063             {
00064                 MultiBoardTutorial.InstructionsBackward();
00065             }
00066             else
00067             {
00068                 if (currentTutorial > 0)
00069                 {
00070                     currentTutorial--;
00071                     SetupCurrentTutorial();
00072                 }
00073             }
00074         }
00075 
00076         public void GoTo(string name)
00077         {
00078             currentTutorial = Tutorials.MyList.IndexOf(name);
00079             SetupCurrentTutorial();
00080         }
00081 
00082         void SetupCurrentTutorial()
00083         {
00084             PreviousButton.SetActive(currentTutorial > 0);
00085             NextButton.SetActive(currentTutorial < Tutorials.MyList.Count - 1);
00086             ResetButton.SetActive(currentTutorial >= 0);
00087             TitleText.text = Tutorials.MyList[currentTutorial];
00088             SceneManager.LoadScene(Tutorials.MyList[currentTutorial]);
00089         }
00090 
00091         public void ToggleSidebar()
00092         {
00093             SidebarAnimator.SetBool("IsVisible", !SidebarAnimator.GetBool("IsVisible"));
00094         }
00095     }
00096 }
```

7.157 SceneKeeper.cs

```
00001 using UnityEngine;
00002 
00003 namespace CardHouse.Tutorial
00004 {
00005     public class SceneKeeper : MonoBehaviour
00006     {
00007         void Start()
00008         {
00009             DontDestroyOnLoad(this);
00010         }
00011     }
00012 }
```

7.158 SceneSpawner.cs

```
00001 using UnityEngine;
00002 using UnityEngine.SceneManagement;
```

```
00003
00004 namespace CardHouse.Tutorial
00005 {
00006     public class SceneSpawner : MonoBehaviour
00007     {
00008         public string SceneToSpawn;
00009         void Start()
00010         {
00011             SceneManager.LoadScene(SceneToSpawn, LoadSceneMode.Additive);
00012         }
00013     }
00014 }
```

7.159 TutorialButton.cs

```
00001 using TMPro;
00002 using UnityEngine;
00003 using UnityEngine.UI;
00004
00005 namespace CardHouse.Tutorial
00006 {
00007     public class TutorialButton : MonoBehaviour
00008     {
00009         public TMP_Text Label;
00010
00011         public void Setup(string text, SandboxManager manager)
00012         {
00013             Label.text = text;
00014             GetComponent<Button>().onClick.AddListener(() => manager.GoTo(text));
00015         }
00016     }
00017 }
```

7.160 StringListScriptable.cs

```
00001 using System.Collections.Generic;
00002 using UnityEngine;
00003
00004 namespace CardHouse.Tutorial
00005 {
00006     public class StringListScriptable : ScriptableObject
00007     {
00008         public List<string> MyList;
00009     }
00010 }
```

Index

Activatable.cs, 251
Activate
    CardHouse.Activatable, 21
    CardHouse.CurrencyCost, 64
    CardHouse.CurrencyOperator, 66
    CardHouse.DiscardCardOperator, 82
    CardHouse.EventChain, 93
    CardHouse.MultiSpriteOperator, 145
    CardHouse.SpriteOperator, 203
    CardHouse.Tutorial.DiscardAllCardsOperator, 81
ActivateAndDelay
    CardHouse.TimedEvent, 218
ActiveButtons
    CardHouse.Phase, 151
ActiveMessage
    CardHouse.CardDropGateDimmer, 42
    CardHouse.DragGateDimmer, 86
    CardHouse.GroupDropGateDimmer, 111
ActiveScene
    CardHouse.Tutorial.LaunchDataScriptable, 131
ActOnTarget
    CardHouse.DiscardTargetCardOperator, 83
    CardHouse.SampleGames.DeckBuilder.DamageGroupOperator, 76
    CardHouse.SampleGames.DeckBuilder.DamageTargetOperator, 77
    CardHouse.Tutorial.WaterPlantAction, 233
AddHoveredGroup
    CardHouse.CardGroup, 45
Adjust
    CardHouse.CurrencyContainer, 62
AdjustArcMargin
    CardHouse.Tutorial.SplayTutorial, 195
AdjustASpadesSlider
    CardHouse.Tutorial.GroupSetupTutorial, 117
AdjustCardLimit
    CardHouse.Tutorial.GridTutorial, 107
AdjustCardsPerRow
    CardHouse.Tutorial.GridTutorial, 107
AdjustCurrencies
    CardHouse.CurrencyRefillOperator, 69
    CardHouse.IncrementCurrencyOperator, 126
AdjustCurrency
    CardHouse.CurrencyRegistry, 70
AdjustDragSwellSlider
    CardHouse.Tutorial.CardDragTutorial, 39
AdjustFlipSpeed
    CardHouse.Tutorial.TransferOperatorTutorialUI, 220
AdjustGrabFrom
    CardHouse.Tutorial.TransferOperatorTutorialUI, 220
AdjustHearts10Slider
    CardHouse.Tutorial.GroupSetupTutorial, 117
AdjustNumberToTransfer
    CardHouse.Tutorial.TransferOperatorTutorialUI, 220
AdjustOffsetX
    CardHouse.Tutorial.CardDragTutorial, 39
AdjustOffsetY
    CardHouse.Tutorial.CardDragTutorial, 39
AdjustQDiamondsSlider
    CardHouse.Tutorial.GroupSetupTutorial, 117
AdjustSeekerGainSlider
    CardHouse.Tutorial.CardDragTutorial, 39
AdjustSendTo
    CardHouse.Tutorial.TransferOperatorTutorialUI, 220
AdjustShuffle
    CardHouse.Tutorial.GroupSetupTutorial, 117
AdjustXOffset
    CardHouse.Tutorial.SplayTutorial, 195
    CardHouse.Tutorial.StackTutorial, 209
AdjustXScale
    CardHouse.Tutorial.GridTutorial, 107
    CardHouse.Tutorial.SplayTutorial, 195
AdjustYOffset
    CardHouse.Tutorial.SplayTutorial, 195
    CardHouse.Tutorial.StackTutorial, 209
AdjustYScale
    CardHouse.Tutorial.GridTutorial, 107
AllGroups
    CardHouse.SampleGames.Solitaire.SolitaireSetup, 192
AllowReset
    CardHouse.SampleGames.Solitaire.SolitaireSetup, 191
AllPhaseDependentButtons
    CardHouse.PhaseManager, 160
AllUnlocked
    CardHouse.GateCollection< T >, 106
Amount
    CardHouse.CurrencyQuantity, 67
AnimCurveFloatSeeker
    CardHouse.AnimCurveFloatSeeker, 23
AnimCurveFloatSeeker.cs, 288
AnimCurveFloatSeekerScriptable.cs, 289
AnimCurveVector3Seeker

CardHouse.AnimCurveVector3Seeker, 26
AnimCurveVector3Seeker.cs, 294
AnimCurveVector3SeekerScriptable.cs, 294
AnyUnlocked
CardHouse.GateCollection< T >, 106
Apply
CardHouse.CardGroupSettings, 50
CardHouse.CurrencyUI, 73
CardHouse.PokerCard, 165
CardHouse.SampleGames.MemoryMatch.MemoryCard, 135
CardHouse.TarotCard, 214
ApplySpacing
CardHouse.CardGridLayout, 43
CardHouse.SlotLayout, 185
CardHouse.SplayLayout, 193
CardHouse.StackLayout, 208
ApplyStrategy
CardHouse.CardGroup, 45
Arcana
CardHouse.ArcanaData, 29
CardHouse.TarotCard, 214
ArcanaData
CardHouse.TarotCard, 215
ArcanaData.cs, 248
ArcanaType
CardHouse.TarotCard, 215
ArcCenterOffset
CardHouse.SplayLayout, 194
ArcMargin
CardHouse.SplayLayout, 194
ArcMarginSlider
CardHouse.Tutorial.SplayTutorial, 196
ArcMarginText
CardHouse.Tutorial.SplayTutorial, 196
ArrivalDistance
CardHouse.ExponentialAngleFloatSeekerScriptable, 97
CardHouse.ExponentialFloatSeeker, 100
CardHouse.ExponentialFloatSeekerScriptable, 101
CardHouse.ExponentialVector3SeekerScriptable, 104
Art
CardHouse.PokerCardDefinition, 167
CardHouse.TarotCardDefinition, 216
ASpadesSlider
CardHouse.Tutorial.GroupSetupTutorial, 117
ASpadesText
CardHouse.Tutorial.GroupSetupTutorial, 117
AttachChildren
CardHouse.SampleGames.Solitaire.SolitaireCardDragHandler, 186

BackArt
CardHouse.CardDefinition, 38
BackButton
CardHouse.Tutorial.MultiBoardTutorial, 141
BackImage
CardHouse.PokerCard, 165
BaseSeekerComponent.cs, 286
BeginDragging
CardHouse.Dragging, 87
BlockAllDrops.cs, 265
ButtonParentScaling
CardHouse.Tutorial.PresentationPointTutorial, 168
ButtonParentTurning
CardHouse.Tutorial.PresentationPointTutorial, 168

CameraPosition
CardHouse.Phase, 151
CanAfford
CardHouse.CurrencyWallet, 74
CanBeUpsideDown
CardHouse.Card, 35
CanBeWatered
CardHouse.Tutorial.Plant, 161
Card
CardHouse.DropParams, 92
CardHouse.SeekerSet, 180
CardHouse.Tutorial.CardDragTutorial, 40
Card.cs, 249
CardBackArt
CardHouse.DeckDefinition, 78
CardCollection
CardHouse.DeckDefinition, 78
CardCount
CardHouse.GroupSetup.GroupPopulationData, 112
CardDefinition.cs, 254
CardDragTutorial.cs, 320
CardDropGateDimmer.cs, 279
CardFacing
CardHouse, 17
CardFacing.cs, 251
CardGridLayout.cs, 274
CardGroup.cs, 267
CardGroupSettings.cs, 275
CardHouse, 15
CardFacing, 17
DragAction, 17
GroupInteractability, 18
GroupName, 18
GroupTargetType, 18
Loyalty, 18
MajorArcanaName, 18
MountingMode, 18
PokerSuit, 18
TarotSuit, 18
CardHouse.Activatable, 21
Activate, 21
OnActivate, 21
CardHouse.AnimCurveFloatSeeker, 22
AnimCurveFloatSeeker, 23
Duration, 24
IsDone, 23
MakeCopy, 23
ProgressCurve, 24

Pump, 23
Timer, 24
CardHouse.AnimCurveFloatSeekerScriptable, 24
Duration, 25
GetStrategy, 25
ProgressCurve, 25
CardHouse.AnimCurveVector3Seeker, 25
AnimCurveVector3Seeker, 26
Duration, 27
IsDone, 27
MakeCopy, 27
ProgressCurve, 27
Pump, 27
Timer, 27
CardHouse.AnimCurveVector3SeekerScriptable, 28
Duration, 28
GetStrategy, 28
ProgressCurve, 28
CardHouse.ArcanaData, 29
Arcana, 29
Rank, 29
Suit, 29
CardHouse.BaseSeekerComponent< T >, 30
IsSeeking, 31
MyStrategy, 31
StartSeeking, 30
Strategy, 31
UseLocalSpace, 31
CardHouse.BlockAllDrops, 31
IsUnlockedInternal, 32
CardHouse.Card, 32
CanBeUpsideDown, 35
FaceHoming, 35
FaceScaling, 35
FaceTurning, 35
Facing, 37
FlipAnimator, 35
GetDiscardGroup, 33
Group, 35
GroupTransitionEvents, 35
HandlePlayed, 33
Homing, 37
IsUpsideDown, 37
OnCardFocused, 36
OnFlipDown, 36
OnFlipUp, 36
OnMount, 36
OnPlay, 36
RootToRotateWhenUpsideDown, 36
Scaling, 37
SetFacing, 34
SetFocus, 34
SetUpsideDown, 34
ToggleFocus, 34
TriggerMountEvents, 34
TriggerUnMountEvents, 34
Turning, 37
UpsideDownChance, 36
CardHouse.Card.GroupTransitionEvent, 121
EntryEvent, 121
ExitEvent, 121
Group, 121
CardHouse.CardDefinition, 37
BackArt, 38
CardHouse.CardDropGateDimmer, 41
ActiveMessage, 42
Handler, 42
InactiveMessage, 42
CardHouse.CardGridLayout, 42
ApplySpacing, 43
CardsPerRow, 44
MarginalCardOffset, 44
Straighten, 44
CardHouse.CardGroup, 44
AddHoveredGroup, 45
ApplyStrategy, 45
DropGates, 48
Get, 45, 46
GetActiveCard, 46
GetClosestMountedCardIndex, 46
HandleTriggerEnter2D, 46
HandleTriggerExit2D, 46
HasRoom, 46
Hilight, 48
HilightedGroup, 49
HilightOnCardEntry, 48
IndexOf, 46
Mount, 47
MountedCards, 48
OnCardUsedOnTarget, 49
OnGroupChanged, 49
OnNewActiveGroup, 49
RemoveHoveredGroup, 47
SafeMount, 47
SetHilightState, 47
Shuffle, 47
ShuffleIn, 47
ShuffleStrategy, 49
UnMount, 48
CardHouse.CardGroupSettings, 50
Apply, 50
CardLimit, 50
DragMountingMode, 50
ForcedFacing, 51
ForcedInteractability, 51
MountedCardAltitude, 51
UseMyScale, 51
CardHouse.CardLoyalty, 51
PlayerIndex, 52
CardHouse.CardSetup, 52
CardHouse.CardTargetCardOperator, 53
DiscardSeekers, 54
MyCard, 54
OnActivate, 53
Target, 54
CardHouse.CardTransferOperator, 54

FlipSpeed, 55
GrabFrom, 55
NumberToTransfer, 55
OnActivate, 55
OnSourceDepletedEventChain, 55
PopPushHomingOverride, 56
SendTo, 56
Transition, 56
TryAgainAfterSourceDepleted, 56
CardHouse.ClickDetector, 56
ClickGates, 57
OnButtonClicked, 57
OnPress, 57
CardHouse.ContinuousInstantVector3Seeker, 58
IsDone, 59
MakeCopy, 59
Pump, 59
CardHouse.ContinuousInstantVector3SeekerScriptable, 59
GetStrategy, 60
CardHouse.CurrencyChangeDetector, 61
OnCurrencyChange, 61
CardHouse.CurrencyContainer, 62
Adjust, 62
Clone, 62
HasMax, 63
HasMin, 63
Max, 63
Min, 63
RefillValue, 63
CardHouse.CurrencyCost, 64
Activate, 64
Cost, 64
CardHouse.CurrencyCost.CostWithLabel, 60
Cost, 60
Label, 60
CardHouse.CurrencyGate, 65
IsUnlockedInternal, 65
CardHouse.CurrencyOperator, 66
Activate, 66
MyRegistry, 66
CardHouse.CurrencyQuantity, 67
Amount, 67
Clone, 67
CurrencyType, 67
CardHouse.CurrencyRefillOperator, 68
AdjustCurrencies, 69
CurrenciesToRefill, 69
CardHouse.CurrencyRegistry, 69
AdjustCurrency, 70
CurrencyDisplayParent, 71
CurrencyDisplayPrefab, 71
CurrentPlayerLabel, 71
GetCurrency, 70
Instance, 71
OnCurrencyChanged, 71
PlayerWallets, 71
Refill, 70
CardHouse.CurrencyScriptable, 72
Name, 72
Sprite, 72
CardHouse.CurrencyUI, 72
Apply, 73
Image, 73
Text, 73
CardHouse.CurrencyWallet, 73
CanAfford, 74
Clone, 74
Currencies, 74
FindCurrency, 74
CardHouse.DeckDefinition, 78
CardBackArt, 78
CardCollection, 78
CardHouse.DeckSetup, 79
CardPrefab, 79
Deck, 79
DeckDefinition, 79
DoSetup, 79
OnSetupCompleteEventChain, 80
RunOnStart, 80
CardHouse.DiscardCardOperator, 81
Activate, 82
CardHouse.DiscardTargetCardOperator, 82
ActOnTarget, 83
TargetDiscardSeekers, 83
CardHouse.DragDetector, 83
DragGates, 84
GroupDropGates, 84
OnDragEnd, 84
OnDragStart, 84
TargetCardGates, 84
CardHouse.DragGateDimmer, 85
ActiveMessage, 86
Handler, 86
InactiveMessage, 86
UpdateHandler, 86
CardHouse.Dragging, 86
BeginDragging, 87
CardPopupDistance, 88
DefaultCardZ, 88
DragHomingStrategy, 88
GetTarget, 87
GrabOffset, 88
Instance, 88
OnDrag, 88
OnDrop, 89
PostDrop, 89
SetNewOffsetOnGrab, 89
StopDragging, 87
UpdateStrategy, 88
UseGrabOffset, 89
CardHouse.DragOperator, 89
DragAction, 90
DragSwell, 90
MyDragDetector, 90
PointUpWhenDragged, 90

PresentationSeekers, 91
SetDragState, 90
CardHouse.DragTransition, 91
DragAction, 91
CardHouse.DropParams, 92
Card, 92
DragType, 92
Source, 92
Target, 92
CardHouse.EventChain, 93
Activate, 93
Events, 93
OnChainFinished, 93
CardHouse.ExponentialAngleFloatSeeker, 95
ExponentialAngleFloatSeeker, 96
Pump, 96
CardHouse.ExponentialAngleFloatSeekerScriptable, 97
ArrivalDistance, 97
Gain, 97
GetStrategy, 97
CardHouse.ExponentialFloatSeeker, 98
ArrivalDistance, 100
ExponentialFloatSeeker, 99
Gain, 100
IsDone, 99
MakeCopy, 99
Pump, 99
CardHouse.ExponentialFloatSeekerScriptable, 100
ArrivalDistance, 101
Gain, 101
GetStrategy, 101
CardHouse.ExponentialVector3Seeker, 101
ExponentialVector3Seeker, 102
IsDone, 102
MakeCopy, 102
Pump, 102
CardHouse.ExponentialVector3SeekerScriptable, 103
ArrivalDistance, 104
GetStrategy, 103
XYGain, 104
ZGain, 104
CardHouse.Gate< T >, 104
IsUnlocked, 105
CardHouse.GateCollection< T >, 105
AllUnlocked, 106
AnyUnlocked, 106
Gates, 106
CardHouse.GroupConditional, 109
MyCard, 110
OnActivate, 110
Responses, 110
CardHouse.GroupDropGateDimmer, 110
ActiveMessage, 111
Handler, 111
InactiveMessage, 111
CardHouse.GroupNameUnityActionKvp, 111
Key, 112
Value, 112
CardHouse.GroupRegistry, 113
Get, 114
GetGroupName, 114
GetLoyalty, 114
GetOwnerIndex, 114
Groups, 114
Instance, 114
CardHouse.GroupRegistry.NamedGroup, 147
Group, 147
Name, 147
PlayerIndex, 147
CardHouse.GroupSetup, 115
DoSetup, 115
GroupPopulationList, 115
GroupsToShuffle, 115
OnSetupCompleteEventChain, 116
RunOnStart, 116
CardHouse.GroupSetup.GroupPopulationData, 112
CardCount, 112
CardPrefab, 112
Group, 112
CardHouse.GroupTransition, 119
Destination, 119
Source, 119
CardHouse.Homing, 123
GetCurrentValue, 124
GetDefaultSeeker, 124
SetNewValue, 124
CardHouse.HoverDetector, 124
OnHover, 125
OnUnHover, 125
CardHouse.IncrementCurrencyOperator, 125
AdjustCurrencies, 126
CurrenciesToChange, 126
CardHouse.InstantFloatSeeker, 127
IsDone, 127
MakeCopy, 127
Pump, 128
CardHouse.InstantVector3Seeker, 128
IsDone, 129
MakeCopy, 129
Pump, 129
CardHouse.LifetimeDestructor, 131
Lifetime, 132
CardHouse.LoyaltyGateCardDrop, 132
Destinations, 133
IsUnlockedInternal, 133
Loyalty, 133
CardHouse.MountDetector, 139
OnMount, 139
CardHouse.MultiplayerBoardSetup, 143
GetAllBoards, 144
PlayerBoard, 144
PlayerCount, 144
PlayerToPlayerInteractions, 144
PvpMode, 144
RunOnStart, 144
Setup, 144

SpacingMultiplier, 145
CardHouse.MultiplayerBoardSetup.GroupTransitionByName, 120
Destination, 120
DragAction, 120
Mode, 120
PhaseIndex, 120
Source, 120
CardHouse.MultiSpriteOperator, 145
Activate, 145
Remove, 146
RemoveVote, 146
SpriteOperators, 146
CardHouse.NoParams, 148
CardHouse.Phase, 150
ActiveButtons, 151
CameraPosition, 151
CardPresentationPosition, 151
End, 150
IsValidDrag, 150
IsValidDragStart, 150
Name, 151
OnPhaseEndEventChain, 151
OnPhaseStartEventChain, 151
PlayerIndex, 151
Start, 150
ValidClickTargets, 152
ValidDrags, 152
CardHouse.PhaseChangeDetector, 152
OnPhaseChange, 153
CardHouse.PhaseConditional, 153
OnActivate, 154
Responses, 154
CardHouse.PhaseGateCardDragStart, 154
IsUnlockedInternal, 155
CardHouse.PhaseGateCardDrop, 155
IsUnlockedInternal, 156
CardHouse.PhaseGateClick, 156
IsUnlockedInternal, 157
CardHouse.PhaseManager, 158
AllPhaseDependentButtons, 160
CurrentPhase, 160
HardReset, 159
Instance, 160
IsValidClick, 159
IsValidDrag, 159
IsValidDragStart, 159
NextPhase, 159
OnPhaseChanged, 160
Phases, 160
PlayerIndex, 160
SetCameraPosition, 159
CardHouse.PokerCard, 164
Apply, 165
BackImage, 165
Image, 165
Rank, 166
Suit, 166
CardHouse.PokerCardDefinition, 166
Art, 167
Rank, 167
Suit, 167
CardHouse.RandomizedCurveVector3SeekerScriptable, 170
GetStrategy, 171
TweakMagnitudeMax, 171
TweakMagnitudeMin, 171
CardHouse.SampleGames, 19
CardHouse.SampleGames.DeckBuilder, 19
CardHouse.SampleGames.DeckBuilder.DamageGroupOperator, 75
ActOnTarget, 76
Damage, 76
CardHouse.SampleGames.DeckBuilder.DamageTargetOperator, 76
ActOnTarget, 77
Damage, 78
CardHouse.SampleGames.DeckBuilder.Health, 122
Change, 122
HealthLevel, 122
HealthText, 122
OnDeath, 122
CardHouse.SampleGames.MemoryMatch, 19
CardHouse.SampleGames.MemoryMatch.MemoryCard, 134
Apply, 135
MySprite, 135
MySpriteRenderer, 135
OnFlippedUp, 135
CardHouse.SampleGames.MemoryMatch.MemoryGame, 135
CardPrefab, 136
Flip, 136
Instance, 136
MatchEffect, 137
MyUI, 137
Restart, 136
Slots, 137
Sprites, 137
CardHouse.SampleGames.MemoryMatch.MemoryUI, 137
MatchText, 138
TimerText, 138
UpdateMatches, 138
UpdateTimer, 138
CardHouse.SampleGames.Solitaire, 19
CardHouse.SampleGames.Solitaire.SolitaireCardDragHandler, 185
AttachChildren, 186
DetatchChildren, 186
CardHouse.SampleGames.Solitaire.SolitaireColumnChangeHandler, 186
Refresh, 187
CardHouse.SampleGames.Solitaire.SolitaireColumnDropGate, 187
IsUnlockedInternal, 188

CardHouse.SampleGames.Solitaire.SolitaireDeckClickHandler, 188
DealCardHandler, 189
FlipHandler, 189
FlipOrReset, 188
MoveToDeckHandler, 189
ResetEventChain, 189
ShuffleHandler, 189
CardHouse.SampleGames.Solitaire.SolitaireScorePileDropGate, 189
IsUnlockedInternal, 190
CardHouse.SampleGames.Solitaire.SolitaireSetup, 190
AllGroups, 192
AllowReset, 191
Columns, 192
DealCards, 191
DealingStrategy, 192
Deck, 192
PreventReset, 191
ResetBoardEventChain, 192
TryResetBoard, 191
CardHouse.SampleGames.Tarot, 19
CardHouse.SampleGames.Tarot.SpreadManager, 197
DealNextCard, 198
Deck, 198
Key, 198
NextSpread, 198
PreviousSpread, 198
ShuffleCardsBackIn, 198
SpreadLabel, 199
SpreadOrderLabelPrefab, 199
Spreads, 199
CardHouse.SampleGames.Tarot.TarotSpread, 216
FillNext, 216
Instructions, 217
Name, 217
Slots, 217
CardHouse.Scaling, 174
GetCurrentValue, 175
GetDefaultSeeker, 175
SetNewValue, 175
CardHouse.Seeker< T >, 177
End, 178
Start, 178
StartSeeking, 177
CardHouse.SeekerScriptable< T >, 178
CardHouse.SeekerScriptableSet, 178
Homing, 179
Scaling, 179
Turning, 179
CardHouse.SeekerSet, 179
Card, 180
FlipSpeed, 180
Homing, 180
Scaling, 180
Turning, 180
CardHouse.SeekerSetList, 180
GetSeekerSetFor, 181
CardHouse.ShuffleOperator, 183
Deck, 184
GroupsToShuffleIntoDeck, 184
OnActivate, 184
CardHouse.SlotLayout, 184
ApplySpacing, 185
CardHouse.SplayLayout, 192
ApplySpacing, 193
ArcCenterOffset, 194
ArcMargin, 194
MarginalCardOffset, 194
CardHouse.SpriteColorOperator, 199
ChangeSprite, 200
Colors, 201
CardHouse.SpriteColorOperator.NamedColor, 146
Color, 147
Name, 147
CardHouse.SpriteImageOperator, 201
ChangeSprite, 202
Sprites, 202
CardHouse.SpriteImageOperator.NamedSprite, 148
Name, 148
Sprite, 148
CardHouse.SpriteOperator, 202
Activate, 203
FavoredState, 204
Remove, 203
SpriteTarget, 204
CardHouse.StackLayout, 207
ApplySpacing, 208
MarginalCardOffset, 208
SecondaryCollider, 208
Straighten, 208
CardHouse.StringUnityActionKvp, 212
Key, 212
Value, 212
CardHouse.TargetCardParams, 213
Source, 213
Target, 213
CardHouse.TarotCard, 213
Apply, 214
Arcana, 214
ArcanaData, 215
ArcanaType, 215
Image, 215
CardHouse.TarotCardDefinition, 215
Art, 216
Data, 216
CardHouse.TestScenes, 20
CardHouse.TestScenes.WaypointTesterCard, 244
Test, 244
WaypointSeekers, 244
CardHouse.TestScenes.WaypointTesterGroup, 245
Test, 245
Waypoints, 245
CardHouse.TimedEvent, 217
ActivateAndDelay, 218
Duration, 218

Event, 218
ExecuteChain, 218
CardHouse.Toggleable, 218
IsActive, 219
SetIsActive, 219
CardHouse.TriggerEnterRelay, 222
Relay, 222
CardHouse.Turning, 223
GetCurrentValue, 224
GetDefaultSeeker, 224
SetNewValue, 224
CardHouse.Tutorial, 20
CardHouse.Tutorial.CardDragTutorial, 38
AdjustDragSwellSlider, 39
AdjustOffsetX, 39
AdjustOffsetY, 39
AdjustSeekerGainSlider, 39
Card, 40
DragSwellSlider, 40
DragSwellText, 40
GrabOffsetToggle, 40
OnGrabOffsetToggled, 39
SeekerGainSlider, 40
SeekerGainText, 40
ShowSwellOutline, 39
XOffsetSlider, 40
XOffsetText, 40
YOffsetSlider, 41
YOffsetText, 41
CardHouse.Tutorial.ClosestCardHighlighter, 58
CardHouse.Tutorial.DiscardAllCardsOperator, 80
Activate, 81
DiscardSeekers, 81
TargetDiscardSeekers, 81
CardHouse.Tutorial.EventChainsTutorial, 94
Chaining, 94
Dropdown, 94
NoChaining, 94
SafeChaining, 95
StartTransition, 94
CardHouse.Tutorial.GridTutorial, 106
AdjustCardLimit, 107
AdjustCardsPerRow, 107
AdjustXScale, 107
AdjustYScale, 107
CardLimitSlider, 108
CardLimitText, 108
CardsPerRowSlider, 108
CardsPerRowText, 108
Deck, 108
Grid, 108
XScaleSlider, 108
XScaleText, 108
YScaleSlider, 109
YScaleText, 109
CardHouse.Tutorial.GroupSetupTutorial, 116
AdjustASpadesSlider, 117
AdjustHearts10Slider, 117
AdjustQDiamondsSlider, 117
AdjustShuffle, 117
ASpadesSlider, 117
ASpadesText, 117
Deck, 118
Hearts10Slider, 118
Hearts10Text, 118
PullBackOperator, 118
QDiamondsSlider, 118
QDiamondsText, 118
Setup, 117
SetupComponent, 118
ShuffleToggle, 118
CardHouse.Tutorial.LaunchDataScriptable, 130
ActiveScene, 131
LaunchedTutorial, 131
OpenScenes, 131
CardHouse.Tutorial.MatureCropDragGate, 133
IsUnlockedInternal, 134
CardHouse.Tutorial.MultiBoardTutorial, 139
BackButton, 141
CommonArea, 141
ForwardButton, 141
InstructionIndex, 141
Instructions, 141
InstructionsBackward, 140
InstructionsForward, 140
InstructionsImage, 141
InstructionsRoot, 142
InstructionsText, 142
PageNumberText, 142
PlayerCountLabel, 142
PlayerCountSlider, 142
SetupBoard, 140
SetupButton, 142
SetupScript, 142
SpacingLabel, 142
SpacingSlider, 143
UpdatePlayerCount, 140
UpdateSpacingMultiplier, 141
CardHouse.Tutorial.MultiBoardTutorial.InstructionImagePair, 129
Image, 130
Text, 130
CardHouse.Tutorial.OutLinks, 149
OnPointerClick, 149
Text, 149
CardHouse.Tutorial.PhaseLabelUpdater, 157
PhaseText, 158
UpdatePhaseLabel, 157
CardHouse.Tutorial.Plant, 161
CanBeWatered, 161
CostJewel, 162
CostText, 162
DescriptionText, 162
HideCost, 161
NameText, 162
Payoff, 161

PossiblePlants, 162
Sprite, 162
Value, 163
Water, 162
CardHouse.Tutorial.PlantGrowthScriptable, 163
Stages, 163
CardHouse.Tutorial.PlantMaturityInfo, 164
Description, 164
Name, 164
Sprite, 164
CardHouse.Tutorial.PresentationPointTutorial, 167
ButtonParentScaling, 168
ButtonParentTurning, 168
ParentScaling, 168
ParentTurning, 168
PlayerIndex, 169
Rotate, 168
RotationMax, 169
RotationMin, 169
RotationSteps, 169
Scale, 168
ScaleMax, 169
ScaleMin, 169
ScaleSteps, 169
UpdateCameraPosition, 168
CardHouse.Tutorial.SandboxManager, 171
GoTo, 172
GoToNext, 172
GoToPrevious, 172
MultiBoardTutorial, 173
NextButton, 173
PreviousButton, 173
Reset, 172
ResetButton, 173
SidebarAnimator, 173
Start, 173
TitleText, 174
ToggleSidebar, 173
TutorialButtonPrefab, 174
TutorialListRoot, 174
Tutorials, 174
CardHouse.Tutorial.SceneKeeper, 176
CardHouse.Tutorial.SceneSpawner, 176
SceneToSpawn, 177
CardHouse.Tutorial.SeekerTutorial, 181
HomingDropdown, 182
ScalingDropdown, 182
SeekerKVPs, 182
Stacks, 182
Transfer, 182
TurningDropdown, 182
Waypoint, 182
CardHouse.Tutorial.SplayTutorial, 194
AdjustArcMargin, 195
AdjustXOffset, 195
AdjustXScale, 195
AdjustYOffset, 195
ArcMarginSlider, 196
ArcMarginText, 196
Deck, 196
Reticle, 196
Splay, 196
XOffsetSlider, 196
XOffsetText, 196
XScaleSlider, 196
XScaleText, 197
YOffsetSlider, 197
YOffsetText, 197
CardHouse.Tutorial.SpriteOperatorTutorial, 204
ColorOperator, 205
ImageOperator, 205
Instance, 205
RegisterColorVote, 205
RegisterImageVote, 205
RemoveColorVote, 205
RemoveImageVote, 205
CardHouse.Tutorial.SpriteVoterTutorial, 206
OnColorDropdownUpdated, 206
OnImageDropdownUpdated, 206
OnStart, 207
CardHouse.Tutorial.StackTutorial, 209
AdjustXOffset, 209
AdjustYOffset, 209
Stack, 210
UseColumnPreset, 209
UseCompactDeckPreset, 210
UseDeckPreset, 210
UseRowPreset, 210
XOffsetSlider, 210
XOffsetText, 210
YOffsetSlider, 210
YOffsetText, 211
CardHouse.Tutorial.StringListScriptable, 211
MyList, 211
CardHouse.Tutorial.StringSeekerKVP, 212
Key, 212
Value, 212
CardHouse.Tutorial.TransferOperatorTutorialUI, 219
AdjustFlipSpeed, 220
AdjustGrabFrom, 220
AdjustNumberToTransfer, 220
AdjustSendTo, 220
FlipSpeedSlider, 220
FlipSpeedText, 220
GrabFromDropdown, 221
NumberToTransferSlider, 221
NumberToTransferText, 221
Operator, 221
SendToDropdown, 221
CardHouse.Tutorial.TutorialButton, 224
Label, 225
Setup, 225
CardHouse.Tutorial.ValidDragTutorial, 229
Dropdown01, 230
Dropdown02, 230
Dropdown10, 231

Dropdown11, 231
Dropdown12, 231
GroupA, 231
GroupB, 231
GroupC, 231
GroupD, 231
PhaseManager, 231
UpdateDropdown01, 230
UpdateDropdown02, 230
UpdateDropdown10, 230
UpdateDropdown11, 230
UpdateDropdown12, 230
CardHouse.Tutorial.WaterPlantAction, 232
ActOnTarget, 233
CardHouse.Tutorial.WaterTargetPlantGate, 233
IsUnlockedInternal, 234
CardHouse.TweakVector3Seeker, 225
MakeCopy, 227
Pump, 227
Tweak, 227
TweakMultiplier, 227
TweakVector3Seeker, 227
CardHouse.TweakVector3SeekerScriptable, 228
GetStrategy, 228
Tweak, 229
TweakMultiplier, 229
CardHouse.WaypointCurveFloatAngleSeeker, 234
IsDone, 235
MakeCopy, 235
Pump, 236
WaypointCurveFloatAngleSeeker, 235
CardHouse.WaypointCurveFloatAngleSeekerScriptable, 236
Duration, 237
GetStrategy, 237
ProgressCurve, 237
CardHouse.WaypointCurveFloatSeeker, 237
IsDone, 239
MakeCopy, 239
Pump, 239
WaypointCurveFloatSeeker, 238
CardHouse.WaypointCurveFloatSeekerScriptable, 239
Duration, 240
GetStrategy, 240
ProgressCurve, 240
CardHouse.WaypointCurveVector3Seeker, 240
IsDone, 242
MakeCopy, 242
Pump, 242
WaypointCurveVector3Seeker, 242
CardHouse.WaypointCurveVector3SeekerScriptable, 242
Duration, 243
GetStrategy, 243
ProgressCurve, 243
CardLimit
CardHouse.CardGroupSettings, 50
CardLimitSlider
CardHouse.Tutorial.GridTutorial, 108
CardLimitText
CardHouse.Tutorial.GridTutorial, 108
CardLoyalty.cs, 281
CardPopupDistance
CardHouse.Dragging, 88
CardPrefab
CardHouse.DeckSetup, 79
CardHouse.GroupSetup.GroupPopulationData, 112
CardHouse.SampleGames.MemoryMatch.MemoryGame, 136
CardPresentationPosition
CardHouse.Phase, 151
CardSetup.cs, 298
CardsPerRow
CardHouse.CardGridLayout, 44
CardsPerRowSlider
CardHouse.Tutorial.GridTutorial, 108
CardsPerRowText
CardHouse.Tutorial.GridTutorial, 108
CardTargetCardOperator.cs, 252
CardTransferOperator.cs, 252
Chaining
CardHouse.Tutorial.EventChainsTutorial, 94
Change
CardHouse.SampleGames.DeckBuilder.Health, 122
ChangeSprite
CardHouse.SpriteColorOperator, 200
CardHouse.SpriteImageOperator, 202
ClickDetector.cs, 255
ClickGates
CardHouse.ClickDetector, 57
Clone
CardHouse.CurrencyContainer, 62
CardHouse.CurrencyQuantity, 67
CardHouse.CurrencyWallet, 74
ClosestCardHighlighter.cs, 322
Color
CardHouse.SpriteColorOperator.NamedColor, 147
ColorOperator
CardHouse.Tutorial.SpriteOperatorTutorial, 205
Colors
CardHouse.SpriteColorOperator, 201
Columns
CardHouse.SampleGames.Solitaire.SolitaireSetup, 192
CommonArea
CardHouse.Tutorial.MultiBoardTutorial, 141
ContinuousInstantVector3Seeker.cs, 294
ContinuousInstantVector3SeekerScriptable.cs, 295
Cost
CardHouse.CurrencyCost, 64
CardHouse.CurrencyCost.CostWithLabel, 60
CostJewel
CardHouse.Tutorial.Plant, 162
CostText

CardHouse.Tutorial.Plant, 162
Currencies
CardHouse.CurrencyWallet, 74
CurrenciesToChange
CardHouse.IncrementCurrencyOperator, 126
CurrenciesToRefill
CardHouse.CurrencyRefillOperator, 69
CurrencyChangeDetector.cs, 256
CurrencyContainer.cs, 257
CurrencyCost.cs, 257
CurrencyDisplayParent
CardHouse.CurrencyRegistry, 71
CurrencyDisplayPrefab
CardHouse.CurrencyRegistry, 71
CurrencyGate.cs, 265
CurrencyOperator.cs, 260
CurrencyQuantity.cs, 257
CurrencyRefillOperator.cs, 261
CurrencyRegistry.cs, 258
CurrencyScriptable.cs, 259
CurrencyType
CardHouse.CurrencyQuantity, 67
CurrencyUI.cs, 260
CurrencyWallet.cs, 260
CurrentPhase
CardHouse.PhaseManager, 160
CurrentPlayerLabel
CardHouse.CurrencyRegistry, 71

Damage
CardHouse.SampleGames.DeckBuilder.DamageGroupOperator, 76
CardHouse.SampleGames.DeckBuilder.DamageTargetOperator, 78
DamageGroupOperator.cs, 309
DamageTargetOperator.cs, 309
Data
CardHouse.TarotCardDefinition, 216
DealCardHandler
CardHouse.SampleGames.Solitaire.SolitaireDeckClickHandler, 189
DealCards
CardHouse.SampleGames.Solitaire.SolitaireSetup, 191
DealingStrategy
CardHouse.SampleGames.Solitaire.SolitaireSetup, 192
DealNextCard
CardHouse.SampleGames.Tarot.SpreadManager, 198
Deck
CardHouse.DeckSetup, 79
CardHouse.SampleGames.Solitaire.SolitaireSetup, 192
CardHouse.SampleGames.Tarot.SpreadManager, 198
CardHouse.ShuffleOperator, 184
CardHouse.Tutorial.GridTutorial, 108
CardHouse.Tutorial.GroupSetupTutorial, 118
CardHouse.Tutorial.SplayTutorial, 196
DeckDefinition
CardHouse.DeckSetup, 79
DeckDefinition.cs, 254
DeckSetup.cs, 298
DefaultCardZ
CardHouse.Dragging, 88
Description
CardHouse.Tutorial.PlantMaturityInfo, 164
DescriptionText
CardHouse.Tutorial.Plant, 162
Destination
CardHouse.GroupTransition, 119
CardHouse.MultiplayerBoardSetup.GroupTransitionByName, 120
Destinations
CardHouse.LoyaltyGateCardDrop, 133
DetatchChildren
CardHouse.SampleGames.Solitaire.SolitaireCardDragHandler, 186
DiscardAllCardsOperator.cs, 325
DiscardCardOperator.cs, 253
DiscardSeekers
CardHouse.CardTargetCardOperator, 54
CardHouse.Tutorial.DiscardAllCardsOperator, 81
DiscardTargetCardOperator.cs, 253
DoSetup
CardHouse.DeckSetup, 79
CardHouse.GroupSetup, 115
DragAction
CardHouse, 17
CardHouse.DragOperator, 90
CardHouse.DragTransition, 91
CardHouse.MultiplayerBoardSetup.GroupTransitionByName, 120
DragAction.cs, 261
DragDetector.cs, 262
DragGateDimmer.cs, 281
DragGates
CardHouse.DragDetector, 84
Dragging.cs, 262
DragHomingStrategy
CardHouse.Dragging, 88
DragMountingMode
CardHouse.CardGroupSettings, 50
DragOperator.cs, 263
DragSwell
CardHouse.DragOperator, 90
DragSwellSlider
CardHouse.Tutorial.CardDragTutorial, 40
DragSwellText
CardHouse.Tutorial.CardDragTutorial, 40
DragTransition.cs, 278
DragType
CardHouse.DropParams, 92
Dropdown
CardHouse.Tutorial.EventChainsTutorial, 94
Dropdown01

CardHouse.Tutorial.ValidDragTutorial, 230
Dropdown02
CardHouse.Tutorial.ValidDragTutorial, 230
Dropdown10
CardHouse.Tutorial.ValidDragTutorial, 231
Dropdown11
CardHouse.Tutorial.ValidDragTutorial, 231
Dropdown12
CardHouse.Tutorial.ValidDragTutorial, 231
DropGates
CardHouse.CardGroup, 48
DropParams.cs, 266
Duration
CardHouse.AnimCurveFloatSeeker, 24
CardHouse.AnimCurveFloatSeekerScriptable, 25
CardHouse.AnimCurveVector3Seeker, 27
CardHouse.AnimCurveVector3SeekerScriptable, 28
CardHouse.TimedEvent, 218
CardHouse.WaypointCurveFloatAngleSeekerScriptable, 237
CardHouse.WaypointCurveFloatSeekerScriptable, 240
CardHouse.WaypointCurveVector3SeekerScriptable, 243

End
CardHouse.Phase, 150
CardHouse.Seeker< T >, 178
EntryEvent
CardHouse.Card.GroupTransitionEvent, 121
Event
CardHouse.TimedEvent, 218
EventChain.cs, 264
EventChainsTutorial.cs, 329
Events
CardHouse.EventChain, 93
ExecuteChain
CardHouse.TimedEvent, 218
ExitEvent
CardHouse.Card.GroupTransitionEvent, 121
ExponentialAngleFloatSeeker
CardHouse.ExponentialAngleFloatSeeker, 96
ExponentialAngleFloatSeeker.cs, 289
ExponentialAngleFloatSeekerScriptable.cs, 289
ExponentialFloatSeeker
CardHouse.ExponentialFloatSeeker, 99
ExponentialFloatSeeker.cs, 289
ExponentialFloatSeekerScriptable.cs, 290
ExponentialVector3Seeker
CardHouse.ExponentialVector3Seeker, 102
ExponentialVector3Seeker.cs, 295
ExponentialVector3SeekerScriptable.cs, 295

FaceHoming
CardHouse.Card, 35
FaceScaling
CardHouse.Card, 35
FaceTurning
CardHouse.Card, 35
Facing
CardHouse.Card, 37
FavoredState
CardHouse.SpriteOperator, 204
FillNext
CardHouse.SampleGames.Tarot.TarotSpread, 216
FindCurrency
CardHouse.CurrencyWallet, 74
Flip
CardHouse.SampleGames.MemoryMatch.MemoryGame, 136
FlipAnimator
CardHouse.Card, 35
FlipHandler
CardHouse.SampleGames.Solitaire.SolitaireDeckClickHandler, 189
FlipOrReset
CardHouse.SampleGames.Solitaire.SolitaireDeckClickHandler, 188
FlipSpeed
CardHouse.CardTransferOperator, 55
CardHouse.SeekerSet, 180
FlipSpeedSlider
CardHouse.Tutorial.TransferOperatorTutorialUI, 220
FlipSpeedText
CardHouse.Tutorial.TransferOperatorTutorialUI, 220
ForcedFacing
CardHouse.CardGroupSettings, 51
ForcedInteractability
CardHouse.CardGroupSettings, 51
ForwardButton
CardHouse.Tutorial.MultiBoardTutorial, 141

Gain
CardHouse.ExponentialAngleFloatSeekerScriptable, 97
CardHouse.ExponentialFloatSeeker, 100
CardHouse.ExponentialFloatSeekerScriptable, 101
Gate.cs, 265
GateCollection.cs, 266
Gates
CardHouse.GateCollection< T >, 106
Get
CardHouse.CardGroup, 45, 46
CardHouse.GroupRegistry, 114
GetActiveCard
CardHouse.CardGroup, 46
GetAllBoards
CardHouse.MultiplayerBoardSetup, 144
GetClosestMountedCardIndex
CardHouse.CardGroup, 46
GetCurrency
CardHouse.CurrencyRegistry, 70
GetCurrentValue
CardHouse.Homing, 124

CardHouse.Scaling, 175
CardHouse.Turning, 224
GetDefaultSeeker
CardHouse.Homing, 124
CardHouse.Scaling, 175
CardHouse.Turning, 224
GetDiscardGroup
CardHouse.Card, 33
GetGroupName
CardHouse.GroupRegistry, 114
GetLoyalty
CardHouse.GroupRegistry, 114
GetOwnerIndex
CardHouse.GroupRegistry, 114
GetSeekerSetFor
CardHouse.SeekerSetList, 181
GetStrategy
CardHouse.AnimCurveFloatSeekerScriptable, 25
CardHouse.AnimCurveVector3SeekerScriptable, 28
CardHouse.ContinuousInstantVector3SeekerScriptable, 60
CardHouse.ExponentialAngleFloatSeekerScriptable, 97
CardHouse.ExponentialFloatSeekerScriptable, 101
CardHouse.ExponentialVector3SeekerScriptable, 103
CardHouse.RandomizedCurveVector3SeekerScriptable, 171
CardHouse.TweakVector3SeekerScriptable, 228
CardHouse.WaypointCurveFloatAngleSeekerScriptable, 237
CardHouse.WaypointCurveFloatSeekerScriptable, 240
CardHouse.WaypointCurveVector3SeekerScriptable, 243
GetTarget
CardHouse.Dragging, 87
GoTo
CardHouse.Tutorial.SandboxManager, 172
GoToNext
CardHouse.Tutorial.SandboxManager, 172
GoToPrevious
CardHouse.Tutorial.SandboxManager, 172
GrabFrom
CardHouse.CardTransferOperator, 55
GrabFromDropdown
CardHouse.Tutorial.TransferOperatorTutorialUI, 221
GrabOffset
CardHouse.Dragging, 88
GrabOffsetToggle
CardHouse.Tutorial.CardDragTutorial, 40
Grid
CardHouse.Tutorial.GridTutorial, 108
GridTutorial.cs, 324
Group
CardHouse.Card, 35
CardHouse.Card.GroupTransitionEvent, 121
CardHouse.GroupRegistry.NamedGroup, 147
CardHouse.GroupSetup.GroupPopulationData, 112
GroupA
CardHouse.Tutorial.ValidDragTutorial, 231
GroupB
CardHouse.Tutorial.ValidDragTutorial, 231
GroupC
CardHouse.Tutorial.ValidDragTutorial, 231
GroupConditional.cs, 255
GroupD
CardHouse.Tutorial.ValidDragTutorial, 231
GroupDropGateDimmer.cs, 281
GroupDropGates
CardHouse.DragDetector, 84
GroupInteractability
CardHouse, 18
GroupInteractability.cs, 273
GroupName
CardHouse, 18
GroupName.cs, 273
GroupPopulationList
CardHouse.GroupSetup, 115
GroupRegistry.cs, 273
Groups
CardHouse.GroupRegistry, 114
GroupSetup.cs, 299
GroupSetupTutorial.cs, 321
GroupsToShuffle
CardHouse.GroupSetup, 115
GroupsToShuffleIntoDeck
CardHouse.ShuffleOperator, 184
GroupTargetType
CardHouse, 18
GroupTransition.cs, 278
GroupTransitionEvents
CardHouse.Card, 35

HandlePlayed
CardHouse.Card, 33
Handler
CardHouse.CardDropGateDimmer, 42
CardHouse.DragGateDimmer, 86
CardHouse.GroupDropGateDimmer, 111
HandleTriggerEnter2D
CardHouse.CardGroup, 46
HandleTriggerExit2D
CardHouse.CardGroup, 46
HardReset
CardHouse.PhaseManager, 159
HasMax
CardHouse.CurrencyContainer, 63
HasMin
CardHouse.CurrencyContainer, 63
HasRoom
CardHouse.CardGroup, 46
Health.cs, 309

HealthLevel
CardHouse.SampleGames.DeckBuilder.Health, 122
HealthText
CardHouse.SampleGames.DeckBuilder.Health, 122
Hearts10Slider
CardHouse.Tutorial.GroupSetupTutorial, 118
Hearts10Text
CardHouse.Tutorial.GroupSetupTutorial, 118
HideCost
CardHouse.Tutorial.Plant, 161
Hilight
CardHouse.CardGroup, 48
HilightedGroup
CardHouse.CardGroup, 49
HilightOnCardEntry
CardHouse.CardGroup, 48
Homing
CardHouse.Card, 37
CardHouse.SeekerScriptableSet, 179
CardHouse.SeekerSet, 180
Homing.cs, 287
HomingDropdown
CardHouse.Tutorial.SeekerTutorial, 182
HoverDetector.cs, 278

Image
CardHouse.CurrencyUI, 73
CardHouse.PokerCard, 165
CardHouse.TarotCard, 215
CardHouse.Tutorial.MultiBoardTutorial.InstructionImagePair, 130
ImageOperator
CardHouse.Tutorial.SpriteOperatorTutorial, 205
InactiveMessage
CardHouse.CardDropGateDimmer, 42
CardHouse.DragGateDimmer, 86
CardHouse.GroupDropGateDimmer, 111
IncrementCurrencyOperator.cs, 261
IndexOf
CardHouse.CardGroup, 46
Instance
CardHouse.CurrencyRegistry, 71
CardHouse.Dragging, 88
CardHouse.GroupRegistry, 114
CardHouse.PhaseManager, 160
CardHouse.SampleGames.MemoryMatch.MemoryGame, 136
CardHouse.Tutorial.SpriteOperatorTutorial, 205
InstantFloatSeeker.cs, 290
InstantVector3Seeker.cs, 296
InstructionIndex
CardHouse.Tutorial.MultiBoardTutorial, 141
Instructions
CardHouse.SampleGames.Tarot.TarotSpread, 217
CardHouse.Tutorial.MultiBoardTutorial, 141
InstructionsBackward
CardHouse.Tutorial.MultiBoardTutorial, 140
InstructionsForward
CardHouse.Tutorial.MultiBoardTutorial, 140
InstructionsImage
CardHouse.Tutorial.MultiBoardTutorial, 141
InstructionsRoot
CardHouse.Tutorial.MultiBoardTutorial, 142
InstructionsText
CardHouse.Tutorial.MultiBoardTutorial, 142
IsActive
CardHouse.Toggleable, 219
IsDone
CardHouse.AnimCurveFloatSeeker, 23
CardHouse.AnimCurveVector3Seeker, 27
CardHouse.ContinuousInstantVector3Seeker, 59
CardHouse.ExponentialFloatSeeker, 99
CardHouse.ExponentialVector3Seeker, 102
CardHouse.InstantFloatSeeker, 127
CardHouse.InstantVector3Seeker, 129
CardHouse.WaypointCurveFloatAngleSeeker, 235
CardHouse.WaypointCurveFloatSeeker, 239
CardHouse.WaypointCurveVector3Seeker, 242
IsSeeking
CardHouse.BaseSeekerComponent< T >, 31
IsUnlocked
CardHouse.Gate< T >, 105
IsUnlockedInternal
CardHouse.BlockAllDrops, 32
CardHouse.CurrencyGate, 65
CardHouse.LoyaltyGateCardDrop, 133
CardHouse.PhaseGateCardDragStart, 155
CardHouse.PhaseGateCardDrop, 156
CardHouse.PhaseGateClick, 157
CardHouse.SampleGames.Solitaire.SolitaireColumnDropGate, 188
CardHouse.SampleGames.Solitaire.SolitaireScorePileDropGate, 190
CardHouse.Tutorial.MatureCropDragGate, 134
CardHouse.Tutorial.WaterTargetPlantGate, 234
IsUpsideDown
CardHouse.Card, 37
IsValidClick
CardHouse.PhaseManager, 159
IsValidDrag
CardHouse.Phase, 150
CardHouse.PhaseManager, 159
IsValidDragStart
CardHouse.Phase, 150
CardHouse.PhaseManager, 159

Key
CardHouse.GroupNameUnityActionKvp, 112
CardHouse.SampleGames.Tarot.SpreadManager, 198
CardHouse.StringUnityActionKvp, 212
CardHouse.Tutorial.StringSeekerKVP, 212

Label
CardHouse.CurrencyCost.CostWithLabel, 60
CardHouse.Tutorial.TutorialButton, 225

LaunchDataScriptable.cs, 319
LaunchedTutorial
CardHouse.Tutorial.LaunchDataScriptable, 131
LaunchTutorialOption.cs, 318
Lifetime
CardHouse.LifetimeDestructor, 132
LifetimeDestructor.cs, 279
Loyalty
CardHouse, 18
CardHouse.LoyaltyGateCardDrop, 133
Loyalty.cs, 282
LoyaltyGateCardDrop.cs, 282

MajorArcanaName
CardHouse, 18
MajorArcanaName.cs, 248
MakeCopy
CardHouse.AnimCurveFloatSeeker, 23
CardHouse.AnimCurveVector3Seeker, 27
CardHouse.ContinuousInstantVector3Seeker, 59
CardHouse.ExponentialFloatSeeker, 99
CardHouse.ExponentialVector3Seeker, 102
CardHouse.InstantFloatSeeker, 127
CardHouse.InstantVector3Seeker, 129
CardHouse.TweakVector3Seeker, 227
CardHouse.WaypointCurveFloatAngleSeeker, 235
CardHouse.WaypointCurveFloatSeeker, 239
CardHouse.WaypointCurveVector3Seeker, 242
MarginalCardOffset
CardHouse.CardGridLayout, 44
CardHouse.SplayLayout, 194
CardHouse.StackLayout, 208
MatchEffect
CardHouse.SampleGames.MemoryMatch.MemoryGame, 137
MatchText
CardHouse.SampleGames.MemoryMatch.MemoryUI, 138
MatureCropDragGate.cs, 330
Max
CardHouse.CurrencyContainer, 63
MemoryCard.cs, 310
MemoryGame.cs, 310
MemoryUI.cs, 312
Min
CardHouse.CurrencyContainer, 63
Mode
CardHouse.MultiplayerBoardSetup.GroupTransitionByName, 120
Mount
CardHouse.CardGroup, 47
MountDetector.cs, 277
MountedCardAltitude
CardHouse.CardGroupSettings, 51
MountedCards
CardHouse.CardGroup, 48
MountingMode
CardHouse, 18
MountingMode.cs, 278
MoveToDeckHandler
CardHouse.SampleGames.Solitaire.SolitaireDeckClickHandler, 189
MultiBoardTutorial
CardHouse.Tutorial.SandboxManager, 173
MultiBoardTutorial.cs, 333
MultiplayerBoardSetup.cs, 300
MultiSpriteOperator.cs, 305
MyCard
CardHouse.CardTargetCardOperator, 54
CardHouse.GroupConditional, 110
MyDragDetector
CardHouse.DragOperator, 90
MyList
CardHouse.Tutorial.StringListScriptable, 211
MyRegistry
CardHouse.CurrencyOperator, 66
MySprite
CardHouse.SampleGames.MemoryMatch.MemoryCard, 135
MySpriteRenderer
CardHouse.SampleGames.MemoryMatch.MemoryCard, 135
MyStrategy
CardHouse.BaseSeekerComponent< T >, 31
MyUI
CardHouse.SampleGames.MemoryMatch.MemoryGame, 137

Name
CardHouse.CurrencyScriptable, 72
CardHouse.GroupRegistry.NamedGroup, 147
CardHouse.Phase, 151
CardHouse.SampleGames.Tarot.TarotSpread, 217
CardHouse.SpriteColorOperator.NamedColor, 147
CardHouse.SpriteImageOperator.NamedSprite, 148
CardHouse.Tutorial.PlantMaturityInfo, 164
NameText
CardHouse.Tutorial.Plant, 162
NextButton
CardHouse.Tutorial.SandboxManager, 173
NextPhase
CardHouse.PhaseManager, 159
NextSpread
CardHouse.SampleGames.Tarot.SpreadManager, 198
NoChaining
CardHouse.Tutorial.EventChainsTutorial, 94
NoParams.cs, 266
NumberToTransfer
CardHouse.CardTransferOperator, 55
NumberToTransferSlider
CardHouse.Tutorial.TransferOperatorTutorialUI, 221
NumberToTransferText
CardHouse.Tutorial.TransferOperatorTutorialUI, 221

OnActivate
CardHouse.Activatable, 21
CardHouse.CardTargetCardOperator, 53
CardHouse.CardTransferOperator, 55
CardHouse.GroupConditional, 110
CardHouse.PhaseConditional, 154
CardHouse.ShuffleOperator, 184
OnButtonClicked
CardHouse.ClickDetector, 57
OnCardFocused
CardHouse.Card, 36
OnCardUsedOnTarget
CardHouse.CardGroup, 49
OnChainFinished
CardHouse.EventChain, 93
OnColorDropdownUpdated
CardHouse.Tutorial.SpriteVoterTutorial, 206
OnCurrencyChange
CardHouse.CurrencyChangeDetector, 61
OnCurrencyChanged
CardHouse.CurrencyRegistry, 71
OnDeath
CardHouse.SampleGames.DeckBuilder.Health, 122
OnDrag
CardHouse.Dragging, 88
OnDragEnd
CardHouse.DragDetector, 84
OnDragStart
CardHouse.DragDetector, 84
OnDrop
CardHouse.Dragging, 89
OnFlipDown
CardHouse.Card, 36
OnFlippedUp
CardHouse.SampleGames.MemoryMatch.MemoryCard, 135
OnFlipUp
CardHouse.Card, 36
OnGrabOffsetToggled
CardHouse.Tutorial.CardDragTutorial, 39
OnGroupChanged
CardHouse.CardGroup, 49
OnHover
CardHouse.HoverDetector, 125
OnImageDropdownUpdated
CardHouse.Tutorial.SpriteVoterTutorial, 206
OnMount
CardHouse.Card, 36
CardHouse.MountDetector, 139
OnNewActiveGroup
CardHouse.CardGroup, 49
OnPhaseChange
CardHouse.PhaseChangeDetector, 153
OnPhaseChanged
CardHouse.PhaseManager, 160
OnPhaseEndEventChain
CardHouse.Phase, 151
OnPhaseStartEventChain
CardHouse.Phase, 151
OnPlay
CardHouse.Card, 36
OnPointerClick
CardHouse.Tutorial.OutLinks, 149
OnPress
CardHouse.ClickDetector, 57
OnSetupCompleteEventChain
CardHouse.DeckSetup, 80
CardHouse.GroupSetup, 116
OnSourceDepletedEventChain
CardHouse.CardTransferOperator, 55
OnStart
CardHouse.Tutorial.SpriteVoterTutorial, 207
OnUnHover
CardHouse.HoverDetector, 125
OpenScenes
CardHouse.Tutorial.LaunchDataScriptable, 131
Operator
CardHouse.Tutorial.TransferOperatorTutorialUI, 221
OutLinks.cs, 335

PageNumberText
CardHouse.Tutorial.MultiBoardTutorial, 142
ParentScaling
CardHouse.Tutorial.PresentationPointTutorial, 168
ParentTurning
CardHouse.Tutorial.PresentationPointTutorial, 168
Payoff
CardHouse.Tutorial.Plant, 161
Phase.cs, 283
PhaseChangeDetector.cs, 283
PhaseConditional.cs, 256
PhaseGateCardDragStart.cs, 284
PhaseGateCardDrop.cs, 284
PhaseGateClick.cs, 255
PhaseIndex
CardHouse.MultiplayerBoardSetup.GroupTransitionByName, 120
PhaseLabelUpdater.cs, 330
PhaseManager
CardHouse.Tutorial.ValidDragTutorial, 231
PhaseManager.cs, 284
Phases
CardHouse.PhaseManager, 160
PhaseText
CardHouse.Tutorial.PhaseLabelUpdater, 158
Plant.cs, 331
PlantGrowthScriptable.cs, 331
PlayerBoard
CardHouse.MultiplayerBoardSetup, 144
PlayerCount
CardHouse.MultiplayerBoardSetup, 144
PlayerCountLabel
CardHouse.Tutorial.MultiBoardTutorial, 142
PlayerCountSlider
CardHouse.Tutorial.MultiBoardTutorial, 142

PlayerIndex
CardHouse.CardLoyalty, 52
CardHouse.GroupRegistry.NamedGroup, 147
CardHouse.Phase, 151
CardHouse.PhaseManager, 160
CardHouse.Tutorial.PresentationPointTutorial, 169
PlayerToPlayerInteractions
CardHouse.MultiplayerBoardSetup, 144
PlayerWallets
CardHouse.CurrencyRegistry, 71
PointUpWhenDragged
CardHouse.DragOperator, 90
PokerCard.cs, 247
PokerCardDefinition.cs, 247
PokerSuit
CardHouse, 18
PokerSuit.cs, 247
PopPushHomingOverride
CardHouse.CardTransferOperator, 56
PossiblePlants
CardHouse.Tutorial.Plant, 162
PostDrop
CardHouse.Dragging, 89
PresentationPointTutorial.cs, 332
PresentationSeekers
CardHouse.DragOperator, 91
PreventReset
CardHouse.SampleGames.Solitaire.SolitaireSetup, 191
PreviousButton
CardHouse.Tutorial.SandboxManager, 173
PreviousSpread
CardHouse.SampleGames.Tarot.SpreadManager, 198
ProgressCurve
CardHouse.AnimCurveFloatSeeker, 24
CardHouse.AnimCurveFloatSeekerScriptable, 25
CardHouse.AnimCurveVector3Seeker, 27
CardHouse.AnimCurveVector3SeekerScriptable, 28
CardHouse.WaypointCurveFloatAngleSeekerScriptable, 237
CardHouse.WaypointCurveFloatSeekerScriptable, 240
CardHouse.WaypointCurveVector3SeekerScriptable, 243
PullBackOperator
CardHouse.Tutorial.GroupSetupTutorial, 118
Pump
CardHouse.AnimCurveFloatSeeker, 23
CardHouse.AnimCurveVector3Seeker, 27
CardHouse.ContinuousInstantVector3Seeker, 59
CardHouse.ExponentialAngleFloatSeeker, 96
CardHouse.ExponentialFloatSeeker, 99
CardHouse.ExponentialVector3Seeker, 102
CardHouse.InstantFloatSeeker, 128
CardHouse.InstantVector3Seeker, 129
CardHouse.TweakVector3Seeker, 227
CardHouse.WaypointCurveFloatAngleSeeker, 236
CardHouse.WaypointCurveFloatSeeker, 239
CardHouse.WaypointCurveVector3Seeker, 242
PvpMode
CardHouse.MultiplayerBoardSetup, 144

QDiamondsSlider
CardHouse.Tutorial.GroupSetupTutorial, 118
QDiamondsText
CardHouse.Tutorial.GroupSetupTutorial, 118

RandomizedCurveVector3SeekerScriptable.cs, 296
Rank
CardHouse.ArcanaData, 29
CardHouse.PokerCard, 166
CardHouse.PokerCardDefinition, 167
Refill
CardHouse.CurrencyRegistry, 70
RefillValue
CardHouse.CurrencyContainer, 63
Refresh
CardHouse.SampleGames.Solitaire.SolitaireColumnChangeHandler, 187
RegisterColorVote
CardHouse.Tutorial.SpriteOperatorTutorial, 205
RegisterImageVote
CardHouse.Tutorial.SpriteOperatorTutorial, 205
Relay
CardHouse.TriggerEnterRelay, 222
Remove
CardHouse.MultiSpriteOperator, 146
CardHouse.SpriteOperator, 203
RemoveColorVote
CardHouse.Tutorial.SpriteOperatorTutorial, 205
RemoveHoveredGroup
CardHouse.CardGroup, 47
RemoveImageVote
CardHouse.Tutorial.SpriteOperatorTutorial, 205
RemoveVote
CardHouse.MultiSpriteOperator, 146
Reset
CardHouse.Tutorial.SandboxManager, 172
ResetBoardEventChain
CardHouse.SampleGames.Solitaire.SolitaireSetup, 192
ResetButton
CardHouse.Tutorial.SandboxManager, 173
ResetEventChain
CardHouse.SampleGames.Solitaire.SolitaireDeckClickHandler, 189
Responses
CardHouse.GroupConditional, 110
CardHouse.PhaseConditional, 154
Restart
CardHouse.SampleGames.MemoryMatch.MemoryGame, 136
Reticle
CardHouse.Tutorial.SplayTutorial, 196
RootToRotateWhenUpsideDown

CardHouse.Card, 36
Rotate
CardHouse.Tutorial.PresentationPointTutorial, 168
RotationMax
CardHouse.Tutorial.PresentationPointTutorial, 169
RotationMin
CardHouse.Tutorial.PresentationPointTutorial, 169
RotationSteps
CardHouse.Tutorial.PresentationPointTutorial, 169
RunOnStart
CardHouse.DeckSetup, 80
CardHouse.GroupSetup, 116
CardHouse.MultiplayerBoardSetup, 144

SafeChaining
CardHouse.Tutorial.EventChainsTutorial, 95
SafeMount
CardHouse.CardGroup, 47
SandboxManager.cs, 335
Scale
CardHouse.Tutorial.PresentationPointTutorial, 168
ScaleMax
CardHouse.Tutorial.PresentationPointTutorial, 169
ScaleMin
CardHouse.Tutorial.PresentationPointTutorial, 169
ScaleSteps
CardHouse.Tutorial.PresentationPointTutorial, 169
Scaling
CardHouse.Card, 37
CardHouse.SeekerScriptableSet, 179
CardHouse.SeekerSet, 180
Scaling.cs, 287
ScalingDropdown
CardHouse.Tutorial.SeekerTutorial, 182
SceneKeeper.cs, 336
SceneSpawner.cs, 336
SceneToSpawn
CardHouse.Tutorial.SceneSpawner, 177
SecondaryCollider
CardHouse.StackLayout, 208
Seeker.cs, 292
SeekerGainSlider
CardHouse.Tutorial.CardDragTutorial, 40
SeekerGainText
CardHouse.Tutorial.CardDragTutorial, 40
SeekerKVPs
CardHouse.Tutorial.SeekerTutorial, 182
SeekerScriptable.cs, 293
SeekerScriptableSet.cs, 293
SeekerSet.cs, 293
SeekerSetList.cs, 293
SeekerTutorial.cs, 328
SendTo
CardHouse.CardTransferOperator, 56
SendToDropdown
CardHouse.Tutorial.TransferOperatorTutorialUI, 221
SetCameraPosition
CardHouse.PhaseManager, 159
SetDragState
CardHouse.DragOperator, 90
SetFacing
CardHouse.Card, 34
SetFocus
CardHouse.Card, 34
SetHilightState
CardHouse.CardGroup, 47
SetIsActive
CardHouse.Toggleable, 219
SetNewOffsetOnGrab
CardHouse.Dragging, 89
SetNewValue
CardHouse.Homing, 124
CardHouse.Scaling, 175
CardHouse.Turning, 224
Setup
CardHouse.MultiplayerBoardSetup, 144
CardHouse.Tutorial.GroupSetupTutorial, 117
CardHouse.Tutorial.TutorialButton, 225
SetupBoard
CardHouse.Tutorial.MultiBoardTutorial, 140
SetupButton
CardHouse.Tutorial.MultiBoardTutorial, 142
SetupComponent
CardHouse.Tutorial.GroupSetupTutorial, 118
SetupScript
CardHouse.Tutorial.MultiBoardTutorial, 142
SetUpsideDown
CardHouse.Card, 34
ShowSwellOutline
CardHouse.Tutorial.CardDragTutorial, 39
Shuffle
CardHouse.CardGroup, 47
ShuffleCardsBackIn
CardHouse.SampleGames.Tarot.SpreadManager, 198
ShuffleHandler
CardHouse.SampleGames.Solitaire.SolitaireDeckClickHandler, 189
ShuffleIn
CardHouse.CardGroup, 47
ShuffleOperator.cs, 254
ShuffleStrategy
CardHouse.CardGroup, 49
ShuffleToggle
CardHouse.Tutorial.GroupSetupTutorial, 118
SidebarAnimator
CardHouse.Tutorial.SandboxManager, 173
SlotLayout.cs, 276
Slots
CardHouse.SampleGames.MemoryMatch.MemoryGame, 137
CardHouse.SampleGames.Tarot.TarotSpread, 217
SolitaireCardDragHandler.cs, 312
SolitaireColumnChangeHandler.cs, 313
SolitaireColumnDropGate.cs, 313
SolitaireDeckClickHandler.cs, 313

SolitaireScorePileDropGate.cs, 314
SolitaireSetup.cs, 314
Source
CardHouse.DropParams, 92
CardHouse.GroupTransition, 119
CardHouse.MultiplayerBoardSetup.GroupTransitionByName, 120
CardHouse.TargetCardParams, 213
SpacingLabel
CardHouse.Tutorial.MultiBoardTutorial, 142
SpacingMultiplier
CardHouse.MultiplayerBoardSetup, 145
SpacingSlider
CardHouse.Tutorial.MultiBoardTutorial, 143
Splay
CardHouse.Tutorial.SplayTutorial, 196
SplayLayout.cs, 276
SplayTutorial.cs, 323
SpreadLabel
CardHouse.SampleGames.Tarot.SpreadManager, 199
SpreadManager.cs, 315
SpreadOrderLabelPrefab
CardHouse.SampleGames.Tarot.SpreadManager, 199
Spreads
CardHouse.SampleGames.Tarot.SpreadManager, 199
Sprite
CardHouse.CurrencyScriptable, 72
CardHouse.SpriteImageOperator.NamedSprite, 148
CardHouse.Tutorial.Plant, 162
CardHouse.Tutorial.PlantMaturityInfo, 164
SpriteColorOperator.cs, 306
SpriteImageOperator.cs, 306
SpriteOperator.cs, 307
SpriteOperators
CardHouse.MultiSpriteOperator, 146
SpriteOperatorTutorial.cs, 325
Sprites
CardHouse.SampleGames.MemoryMatch.MemoryGame, 137
CardHouse.SpriteImageOperator, 202
SpriteTarget
CardHouse.SpriteOperator, 204
SpriteVoterTutorial.cs, 326
Stack
CardHouse.Tutorial.StackTutorial, 210
StackLayout.cs, 277
Stacks
CardHouse.Tutorial.SeekerTutorial, 182
StackTutorial.cs, 322
Stages
CardHouse.Tutorial.PlantGrowthScriptable, 163
Start
CardHouse.Phase, 150
CardHouse.Seeker< T >, 178
CardHouse.Tutorial.SandboxManager, 173
StartSeeking
CardHouse.BaseSeekerComponent< T >, 30
CardHouse.Seeker< T >, 177
StartTransition
CardHouse.Tutorial.EventChainsTutorial, 94
StopDragging
CardHouse.Dragging, 87
Straighten
CardHouse.CardGridLayout, 44
CardHouse.StackLayout, 208
Strategy
CardHouse.BaseSeekerComponent< T >, 31
StringListScriptable.cs, 337
Suit
CardHouse.ArcanaData, 29
CardHouse.PokerCard, 166
CardHouse.PokerCardDefinition, 167

Target
CardHouse.CardTargetCardOperator, 54
CardHouse.DropParams, 92
CardHouse.TargetCardParams, 213
TargetCardGates
CardHouse.DragDetector, 84
TargetCardParams.cs, 266
TargetDiscardSeekers
CardHouse.DiscardTargetCardOperator, 83
CardHouse.Tutorial.DiscardAllCardsOperator, 81
TarotCard.cs, 248
TarotCardDefinition.cs, 249
TarotSpread.cs, 316
TarotSuit
CardHouse, 18
TarotSuit.cs, 249
Test
CardHouse.TestScenes.WaypointTesterCard, 244
CardHouse.TestScenes.WaypointTesterGroup, 245
Text
CardHouse.CurrencyUI, 73
CardHouse.Tutorial.MultiBoardTutorial.InstructionImagePair, 130
CardHouse.Tutorial.OutLinks, 149
TimedEvent.cs, 264
Timer
CardHouse.AnimCurveFloatSeeker, 24
CardHouse.AnimCurveVector3Seeker, 27
TimerText
CardHouse.SampleGames.MemoryMatch.MemoryUI, 138
TitleText
CardHouse.Tutorial.SandboxManager, 174
Toggleable.cs, 308
ToggleFocus
CardHouse.Card, 34
ToggleSidebar
CardHouse.Tutorial.SandboxManager, 173
Transfer
CardHouse.Tutorial.SeekerTutorial, 182

TransferOperatorTutorialUI.cs, 327
Transition
CardHouse.CardTransferOperator, 56
TriggerEnterRelay.cs, 279
TriggerMountEvents
CardHouse.Card, 34
TriggerUnMountEvents
CardHouse.Card, 34
TryAgainAfterSourceDepleted
CardHouse.CardTransferOperator, 56
TryResetBoard
CardHouse.SampleGames.Solitaire.SolitaireSetup, 191
Turning
CardHouse.Card, 37
CardHouse.SeekerScriptableSet, 179
CardHouse.SeekerSet, 180
Turning.cs, 288
TurningDropdown
CardHouse.Tutorial.SeekerTutorial, 182
TutorialButton.cs, 337
TutorialButtonPrefab
CardHouse.Tutorial.SandboxManager, 174
TutorialListRoot
CardHouse.Tutorial.SandboxManager, 174
Tutorials
CardHouse.Tutorial.SandboxManager, 174
Tweak
CardHouse.TweakVector3Seeker, 227
CardHouse.TweakVector3SeekerScriptable, 229
TweakMagnitudeMax
CardHouse.RandomizedCurveVector3SeekerScriptable, 171
TweakMagnitudeMin
CardHouse.RandomizedCurveVector3SeekerScriptable, 171
TweakMultiplier
CardHouse.TweakVector3Seeker, 227
CardHouse.TweakVector3SeekerScriptable, 229
TweakVector3Seeker
CardHouse.TweakVector3Seeker, 227
TweakVector3Seeker.cs, 296
TweakVector3SeekerScriptable.cs, 297

UnMount
CardHouse.CardGroup, 48
UpdateCameraPosition
CardHouse.Tutorial.PresentationPointTutorial, 168
UpdateDropdown01
CardHouse.Tutorial.ValidDragTutorial, 230
UpdateDropdown02
CardHouse.Tutorial.ValidDragTutorial, 230
UpdateDropdown10
CardHouse.Tutorial.ValidDragTutorial, 230
UpdateDropdown11
CardHouse.Tutorial.ValidDragTutorial, 230
UpdateDropdown12
CardHouse.Tutorial.ValidDragTutorial, 230
UpdateHandler
CardHouse.DragGateDimmer, 86
UpdateMatches
CardHouse.SampleGames.MemoryMatch.MemoryUI, 138
UpdatePhaseLabel
CardHouse.Tutorial.PhaseLabelUpdater, 157
UpdatePlayerCount
CardHouse.Tutorial.MultiBoardTutorial, 140
UpdateSpacingMultiplier
CardHouse.Tutorial.MultiBoardTutorial, 141
UpdateStrategy
CardHouse.Dragging, 88
UpdateTimer
CardHouse.SampleGames.MemoryMatch.MemoryUI, 138
UpsideDownChance
CardHouse.Card, 36
UseColumnPreset
CardHouse.Tutorial.StackTutorial, 209
UseCompactDeckPreset
CardHouse.Tutorial.StackTutorial, 210
UseDeckPreset
CardHouse.Tutorial.StackTutorial, 210
UseGrabOffset
CardHouse.Dragging, 89
UseLocalSpace
CardHouse.BaseSeekerComponent< T >, 31
UseMyScale
CardHouse.CardGroupSettings, 51
UseRowPreset
CardHouse.Tutorial.StackTutorial, 210
Utils.cs, 308

ValidClickTargets
CardHouse.Phase, 152
ValidDrags
CardHouse.Phase, 152
ValidDragTutorial.cs, 329
Value
CardHouse.GroupNameUnityActionKvp, 112
CardHouse.StringUnityActionKvp, 212
CardHouse.Tutorial.Plant, 163
CardHouse.Tutorial.StringSeekerKVP, 212

Water
CardHouse.Tutorial.Plant, 162
WaterPlantAction.cs, 332
WaterTargetPlantGate.cs, 332
Waypoint
CardHouse.Tutorial.SeekerTutorial, 182
WaypointCurveFloatAngleSeeker
CardHouse.WaypointCurveFloatAngleSeeker, 235
WaypointCurveFloatAngleSeeker.cs, 290
WaypointCurveFloatAngleSeekerScriptable.cs, 291
WaypointCurveFloatSeeker
CardHouse.WaypointCurveFloatSeeker, 238
WaypointCurveFloatSeeker.cs, 291
WaypointCurveFloatSeekerScriptable.cs, 292
WaypointCurveVector3Seeker

CardHouse.WaypointCurveVector3Seeker, 242
WaypointCurveVector3Seeker.cs, 297
WaypointCurveVector3SeekerScriptable.cs, 297
Waypoints
CardHouse.TestScenes.WaypointTesterGroup, 245
WaypointSeekers
CardHouse.TestScenes.WaypointTesterCard, 244
WaypointTesterCard.cs, 317
WaypointTesterGroup.cs, 317

XOffsetSlider
CardHouse.Tutorial.CardDragTutorial, 40
CardHouse.Tutorial.SplayTutorial, 196
CardHouse.Tutorial.StackTutorial, 210
XOffsetText
CardHouse.Tutorial.CardDragTutorial, 40
CardHouse.Tutorial.SplayTutorial, 196
CardHouse.Tutorial.StackTutorial, 210
XScaleSlider
CardHouse.Tutorial.GridTutorial, 108
CardHouse.Tutorial.SplayTutorial, 196
XScaleText
CardHouse.Tutorial.GridTutorial, 108
CardHouse.Tutorial.SplayTutorial, 197
XYGain
CardHouse.ExponentialVector3SeekerScriptable, 104

YOffsetSlider
CardHouse.Tutorial.CardDragTutorial, 41
CardHouse.Tutorial.SplayTutorial, 197
CardHouse.Tutorial.StackTutorial, 210
YOffsetText
CardHouse.Tutorial.CardDragTutorial, 41
CardHouse.Tutorial.SplayTutorial, 197
CardHouse.Tutorial.StackTutorial, 211
YScaleSlider
CardHouse.Tutorial.GridTutorial, 109
YScaleText
CardHouse.Tutorial.GridTutorial, 109

ZGain
CardHouse.ExponentialVector3SeekerScriptable, 104