Prototype

Generated by Doxygen 1.11.0

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

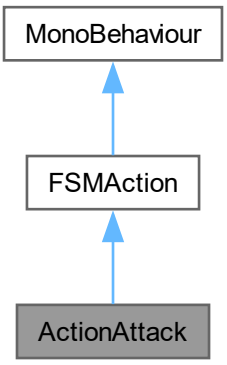
Here is a list of all files with brief descriptions:

Chapter 4

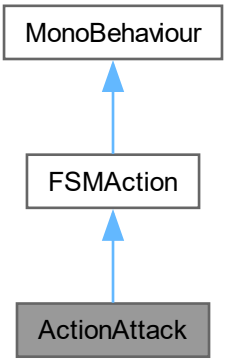
Class Documentation

4.1 ActionAttack Class Reference

Inheritance diagram for ActionAttack:



Collaboration diagram for ActionAttack:



Public Member Functions

- override void Act ()

Public Member Functions inherited from FSMAction

- void Act ()

4.1.1 Detailed Description

Definition at line 6 of file ActionAttack.cs.

4.1.2 Member Function Documentation

4.1.2.1 Act()

```
override void ActionAttack.Act ()
```

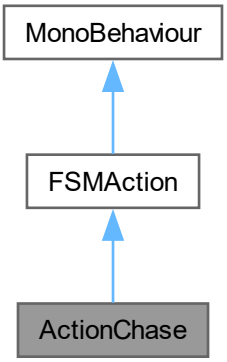
Definition at line 20 of file ActionAttack.cs.

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

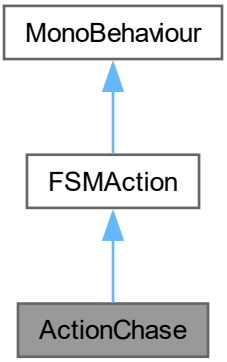
- Enemy/FSM/Actions/ActionAttack.cs

4.2 ActionChase Class Reference

Inheritance diagram for ActionChase:



Collaboration diagram for ActionChase:



Public Member Functions

- override void Act ()

Public Member Functions inherited from FSMAction

- void Act ()

4.2.1 Detailed Description

Definition at line 3 of file ActionChase.cs.

4.2.2 Member Function Documentation

4.2.2.1 Act()

```
override void ActionChase.Act ()
```

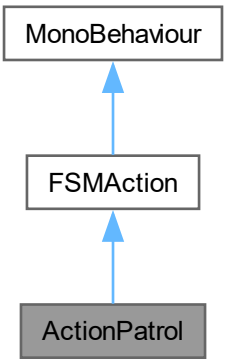
Definition at line 15 of file ActionChase.cs.

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

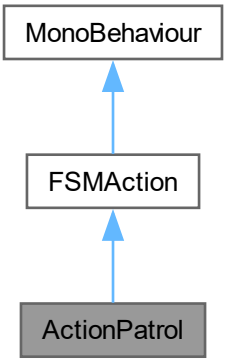
- Enemy/FSM/Actions/ActionChase.cs

4.3 ActionPatrol Class Reference

Inheritance diagram for ActionPatrol:



Collaboration diagram for ActionPatrol:



Public Member Functions

- override void Act ()

Public Member Functions inherited from FSMAction

- void Act ()

4.3.1 Detailed Description

Definition at line 4 of file ActionPatrol.cs.

4.3.2 Member Function Documentation

4.3.2.1 Act()

```
override void ActionPatrol.Act ()
```

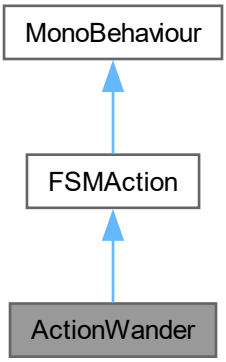
Definition at line 18 of file ActionPatrol.cs.

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

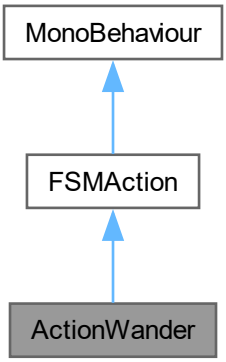
- Enemy/FSM/Actions/ActionPatrol.cs

4.4 ActionWander Class Reference

Inheritance diagram for ActionWander:



Collaboration diagram for ActionWander:



Public Member Functions

- override void Act ()

Public Member Functions inherited from FSMAction

- void Act ()

4.4.1 Detailed Description

Definition at line 4 of file ActionWander.cs.

4.4.2 Member Function Documentation

4.4.2.1 Act()

```
override void ActionWander.Act ()
```

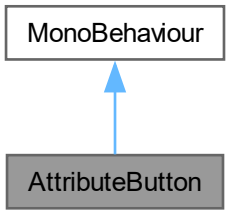
Definition at line 18 of file ActionWander.cs.

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

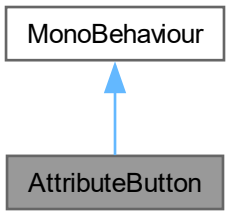
- Enemy/FSM/Actions/ActionWander.cs

4.5 AttributeButton Class Reference

Inheritance diagram for AttributeButton:



Collaboration diagram for AttributeButton:



Public Member Functions

- void SelectAttribute ()

Events

- static Action< AttributeType > OnAttributeSelectedEvent

4.5.1 Detailed Description

Definition at line 6 of file AttributeButton.cs.

4.5.2 Member Function Documentation

4.5.2.1 SelectAttribute()

```
void AttributeButton.SelectAttribute ()
```

Definition at line 13 of file AttributeButton.cs.

4.5.3 Event Documentation

4.5.3.1 OnAttributeSelectedEvent

```
Action<AttributeType> AttributeButton.OnAttributeSelectedEvent  [static]
```

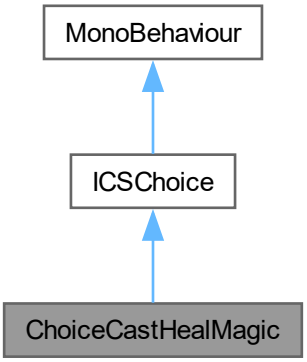
Definition at line 8 of file AttributeButton.cs.

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

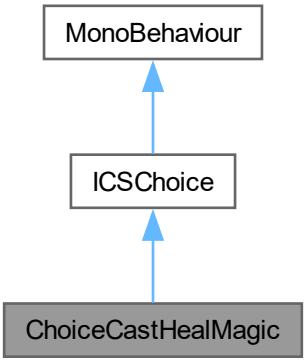
- Extra/AttributeButton.cs

4.6 ChoiceCastHealMagic Class Reference

Inheritance diagram for ChoiceCastHealMagic:



Collaboration diagram for ChoiceCastHealMagic:



Public Member Functions

- void PerformChoice ()

Public Member Functions inherited from ICSChoice

- void EventInvoke ()

Additional Inherited Members

Events inherited from ICSChoice

- static Action ChoicePerformedEvent

4.6.1 Detailed Description

Definition at line 7 of file ChoiceCastHealMagic.cs.

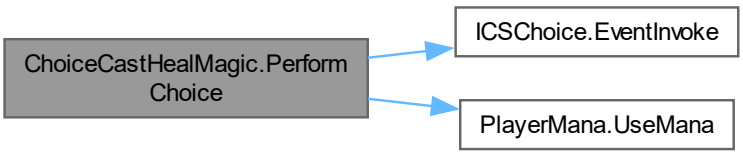
4.6.2 Member Function Documentation

4.6.2.1 PerformChoice()

```
void ChoiceCastHealMagic.PerformChoice ()
```

Definition at line 12 of file ChoiceCastHealMagic.cs.

Here is the call graph for this function:

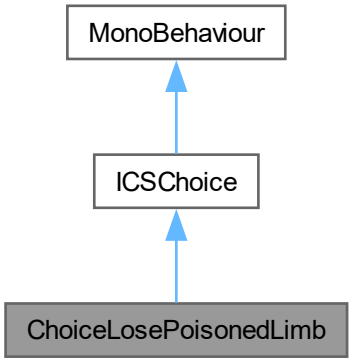


The documentation for this class was generated from the following file:

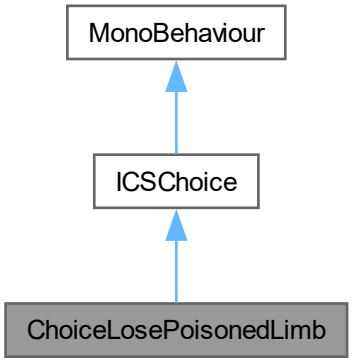
- ICS/Poisoned/ChoiceCastHealMagic.cs

4.7 ChoiceLosePoisonedLimb Class Reference

Inheritance diagram for ChoiceLosePoisonedLimb:



Collaboration diagram for ChoiceLosePoisonedLimb:



Public Member Functions

- void PerformChoice ()

Public Member Functions inherited from ICSChoice

- void EventInvoke ()

Additional Inherited Members

Events inherited from ICSChoice

- static Action ChoicePerformedEvent

4.7.1 Detailed Description

Definition at line 5 of file ChoiceLosePoisonedLimb.cs.

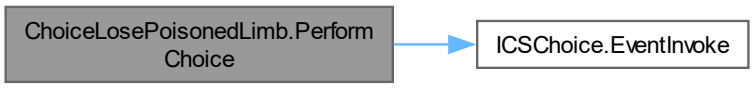
4.7.2 Member Function Documentation

4.7.2.1 PerformChoice()

```
void ChoiceLosePoisonedLimb.PerformChoice ()
```

Definition at line 10 of file ChoiceLosePoisonedLimb.cs.

Here is the call graph for this function:

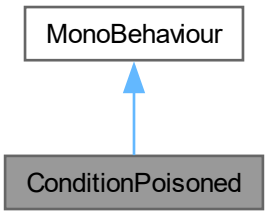


The documentation for this class was generated from the following file:

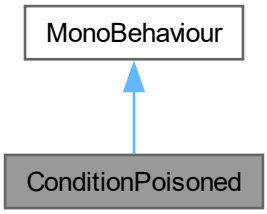
- ICS/Poisoned/ChoiceLosePoisonedLimb.cs

4.8 ConditionPoisoned Class Reference

Inheritance diagram for ConditionPoisoned:



Collaboration diagram for ConditionPoisoned:



Events

- static Action PoisonConditionEvent

4.8.1 Detailed Description

Definition at line 8 of file ConditionPoisoned.cs.

4.8.2 Event Documentation

4.8.2.1 PoisonConditionEvent

```
Action ConditionPoisoned.PoisonConditionEvent  [static]
```

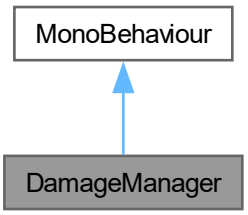
Definition at line 10 of file ConditionPoisoned.cs.

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

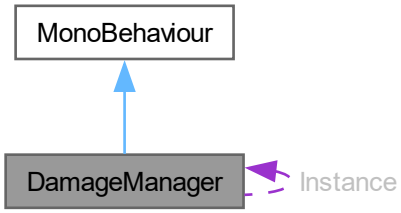
- ICS/Poisoned/ConditionPoisoned.cs

4.9 DamageManager Class Reference

Inheritance diagram for DamageManager:



Collaboration diagram for DamageManager:



Public Member Functions

- void ShowDamageText (float damage, Transform parent)

Static Public Attributes

- static DamageManager Instance

4.9.1 Detailed Description

Definition at line 4 of file DamageManager.cs.

4.9.2 Member Function Documentation

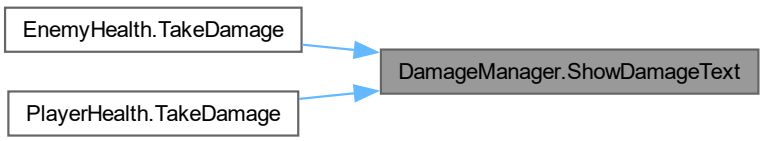
4.9.2.1 ShowDamageText()

```
void DamageManager.ShowDamageText (
            float damage,
            Transform parent)
```

Definition at line 16 of file DamageManager.cs.

Here is the call graph for this function:

Here is the caller graph for this function:



4.9.3 Member Data Documentation

4.9.3.1 Instance

```
DamageManager DamageManager.Instance  [static]
```

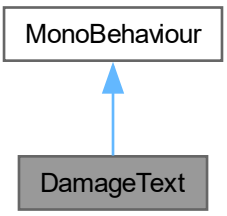
Definition at line 6 of file DamageManager.cs.

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

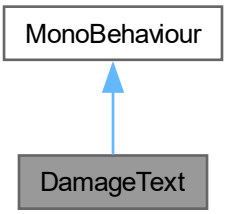
- Managers/DamageManager.cs

4.10 DamageText Class Reference

Inheritance diagram for DamageText:



Collaboration diagram for DamageText:



Public Member Functions

- void SetDamageText (float damage)
- void DestroyDamageText ()

4.10.1 Detailed Description

Definition at line 6 of file DamageText.cs.

4.10.2 Member Function Documentation

4.10.2.1 DestroyDamageText()

```
void DamageText.DestroyDamageText ()
```

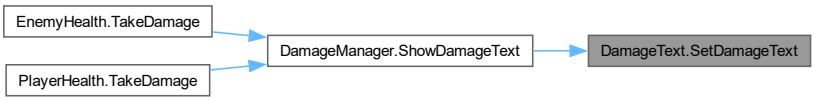
Definition at line 16 of file DamageText.cs.

4.10.2.2 SetDamageText()

```
void DamageText.SetDamageText (
            float damage)
```

Definition at line 11 of file DamageText.cs.

Here is the caller graph for this function:

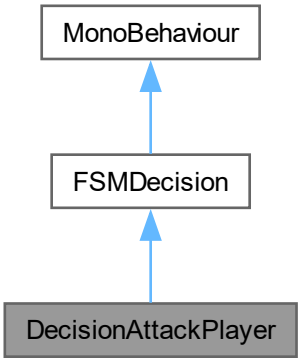


The documentation for this class was generated from the following file:

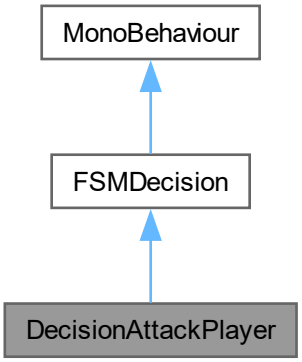
- Text/DamageText.cs

4.11 DecisionAttackPlayer Class Reference

Inheritance diagram for DecisionAttackPlayer:



Collaboration diagram for DecisionAttackPlayer:



Public Member Functions

- override bool Decide ()

Public Member Functions inherited from FSMDecision

- bool Decide ()

4.11.1 Detailed Description

Definition at line 4 of file DecisionAttackPlayer.cs.

4.11.2 Member Function Documentation

4.11.2.1 Decide()

```
override bool DecisionAttackPlayer.Decide ()
```

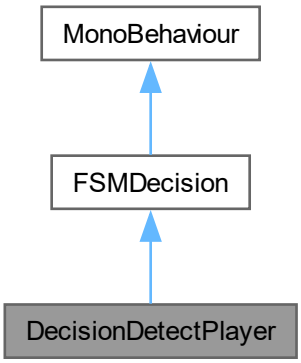
Definition at line 18 of file DecisionAttackPlayer.cs.

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

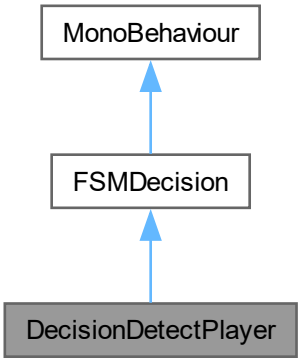
- Enemy/FSM/Decisions/DecisionAttackPlayer.cs

4.12 DecisionDetectPlayer Class Reference

Inheritance diagram for DecisionDetectPlayer:



Collaboration diagram for DecisionDetectPlayer:



Public Member Functions

- override bool Decide ()

Public Member Functions inherited from FSMDecision

- bool Decide ()

4.12.1 Detailed Description

Definition at line 5 of file DecisionDetectPlayer.cs.

4.12.2 Member Function Documentation

4.12.2.1 Decide()

```
override bool DecisionDetectPlayer.Decide ()
```

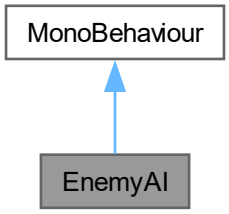
Definition at line 18 of file DecisionDetectPlayer.cs.

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

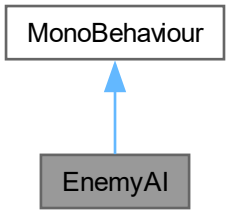
- Enemy/FSM/Decisions/DecisionDetectPlayer.cs

4.13 EnemyAI Class Reference

Inheritance diagram for EnemyAI:



Collaboration diagram for EnemyAI:



Public Member Functions

- void ChangeState (string newStateID)

Properties

- FSMState CurrentState `[get, set]`
- Transform Player `[get, set]`

4.13.1 Detailed Description

Definition at line 4 of file EnemyAI.cs.

4.13.2 Member Function Documentation

4.13.2.1 ChangeState()

```
void EnemyAI.ChangeState (
            string newStateID)
```

Definition at line 22 of file EnemyAI.cs.

4.13.3 Property Documentation

4.13.3.1 CurrentState

```
FSMState EnemyAI.CurrentState  [get], [set]
```

Definition at line 9 of file EnemyAI.cs.

4.13.3.2 Player

```
Transform EnemyAI.Player  [get], [set]
```

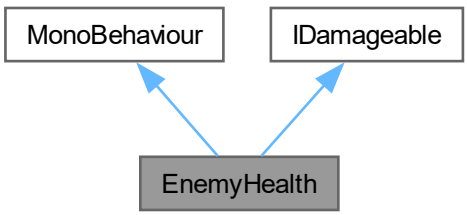
Definition at line 10 of file EnemyAI.cs.

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

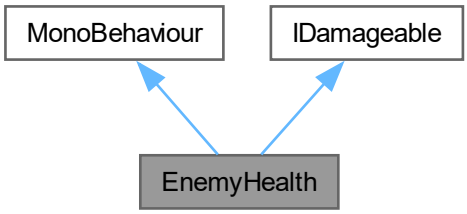
- Enemy/EnemyAI.cs

4.14 EnemyHealth Class Reference

Inheritance diagram for EnemyHealth:



Collaboration diagram for EnemyHealth:



Public Member Functions

- void TakeDamage (float amount)

Properties

- float CurrentHealth `[get, set]`

Events

- static Action OnEnemyDeadEvent

4.14.1 Detailed Description

Definition at line 4 of file EnemyHealth.cs.

4.14.2 Member Function Documentation

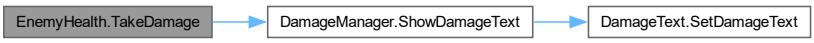
4.14.2.1 TakeDamage()

```
void EnemyHealth.TakeDamage (
            float amount)
```

Implements IDamageable.

Definition at line 32 of file EnemyHealth.cs.

Here is the call graph for this function:



4.14.3 Property Documentation

4.14.3.1 CurrentHealth

```
float EnemyHealth.CurrentHealth  [get], [set]
```

Definition at line 11 of file EnemyHealth.cs.

4.14.4 Event Documentation

4.14.4.1 OnEnemyDeadEvent

```
Action EnemyHealth.OnEnemyDeadEvent  [static]
```

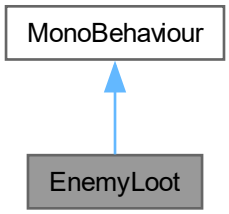
Definition at line 6 of file EnemyHealth.cs.

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

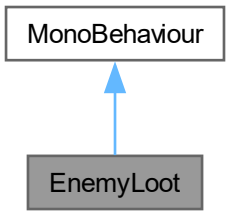
- Enemy/EnemyHealth.cs

4.15 EnemyLoot Class Reference

Inheritance diagram for EnemyLoot:



Collaboration diagram for EnemyLoot:



Properties

- float ExpDrop `[get]`

4.15.1 Detailed Description

Definition at line 5 of file EnemyLoot.cs.

4.15.2 Property Documentation

4.15.2.1 ExpDrop

```
float EnemyLoot.ExpDrop [get]
```

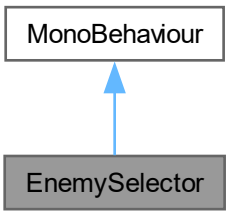
Definition at line 10 of file EnemyLoot.cs.

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

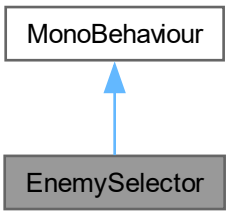
- Enemy/EnemyLoot.cs

4.16 EnemySelector Class Reference

Inheritance diagram for EnemySelector:



Collaboration diagram for EnemySelector:



Public Member Functions

- void NoSelectionCallBack ()

4.16.1 Detailed Description

Definition at line 5 of file EnemySelector.cs.

4.16.2 Member Function Documentation

4.16.2.1 NoSelectionCallBack()

```
void EnemySelector.NoSelectionCallBack ()
```

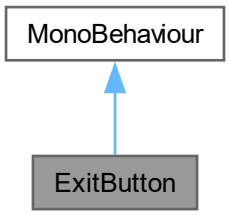
Definition at line 22 of file EnemySelector.cs.

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

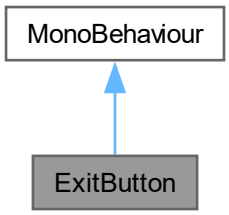
- Enemy/EnemySelector.cs

4.17 ExitButton Class Reference

Inheritance diagram for ExitButton:



Collaboration diagram for ExitButton:



Public Member Functions

- void ExitGame ()

4.17.1 Detailed Description

Definition at line 6 of file ExitButton.cs.

4.17.2 Member Function Documentation

4.17.2.1 ExitGame()

```
void ExitButton.ExitGame ()
```

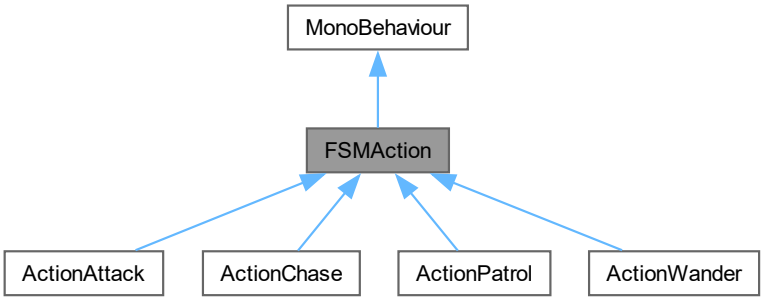
Definition at line 8 of file ExitButton.cs.

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

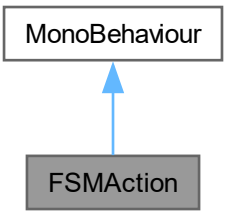
- Extra/ExitButton.cs

4.18 FSMAction Class Reference

Inheritance diagram for FSMAction:



Collaboration diagram for FSMAction:



Public Member Functions

- void Act ()

4.18.1 Detailed Description

Definition at line 3 of file FSMAction.cs.

4.18.2 Member Function Documentation

4.18.2.1 Act()

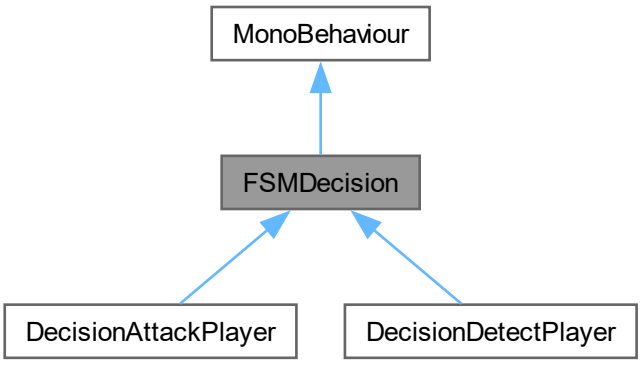
```
void FSMAction.Act ()  [abstract]
```

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

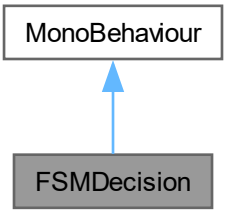
- Enemy/FSM/FSMAction.cs

4.19 FSMDecision Class Reference

Inheritance diagram for FSMDecision:



Collaboration diagram for FSMDecision:



Public Member Functions

- bool Decide ()

4.19.1 Detailed Description

Definition at line 3 of file FSMDecision.cs.

4.19.2 Member Function Documentation

4.19.2.1 Decide()

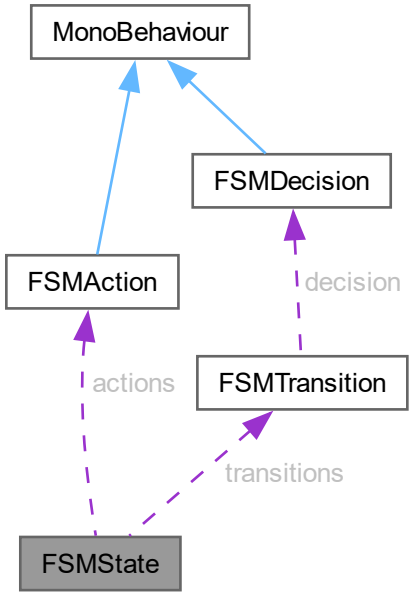
```
bool FSMDecision.Decide ()  [abstract]
```

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

- Enemy/FSM/FSMDecision.cs

4.20 FSMState Class Reference

Collaboration diagram for FSMState:



Public Member Functions

- void UpdateState (EnemyAI enemyAI)

Public Attributes

- string id
- FSMAction[ ] actions
- FSMTransition[ ] transitions

4.20.1 Detailed Description

Definition at line 4 of file FSMState.cs.

4.20.2 Member Function Documentation

4.20.2.1 UpdateState()

```
void FSMState.UpdateState (
            EnemyAI enemyAI)
```

Definition at line 10 of file FSMState.cs.

4.20.3 Member Data Documentation

4.20.3.1 actions

```
FSMAction [] FSMState.actions
```

Definition at line 7 of file FSMState.cs.

4.20.3.2 id

```
string FSMState.id
```

Definition at line 6 of file FSMState.cs.

4.20.3.3 transitions

```
FSMTransition [] FSMState.transitions
```

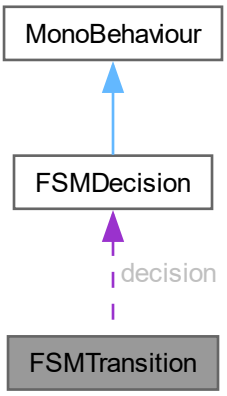
Definition at line 8 of file FSMState.cs.

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

- Enemy/FSM/FSMState.cs

4.21 FSMTransition Class Reference

Collaboration diagram for FSMTransition:



Public Attributes

- FSMDecision decision
- string trueState
- string falseState

4.21.1 Detailed Description

Definition at line 5 of file FSMTransition.cs.

4.21.2 Member Data Documentation

4.21.2.1 decision

```
FSMDecision FSMTransition.decision
```

Definition at line 7 of file FSMTransition.cs.

4.21.2.2 falseState

```
string FSMTransition.falseState
```

Definition at line 9 of file FSMTransition.cs.

4.21.2.3 trueState

```
string FSMTransition.trueState
```

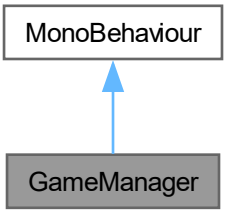
Definition at line 8 of file FSMTransition.cs.

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

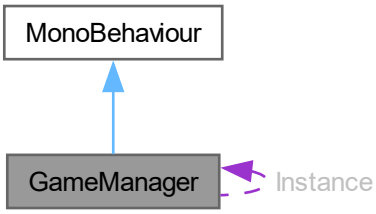
- Enemy/FSM/FSMTransition.cs

4.22 GameManager Class Reference

Inheritance diagram for GameManager:



Collaboration diagram for GameManager:



Public Member Functions

- void AddPlayerExp (float expAmount)

Static Public Attributes

- static GameManager Instance

4.22.1 Detailed Description

Definition at line 4 of file GameManager.cs.

4.22.2 Member Function Documentation

4.22.2.1 AddPlayerExp()

```
void GameManager.AddPlayerExp (
            float expAmount)
```

Definition at line 23 of file GameManager.cs.

Here is the call graph for this function:

4.22.3 Member Data Documentation

4.22.3.1 Instance

```
GameManager GameManager.Instance  [static]
```

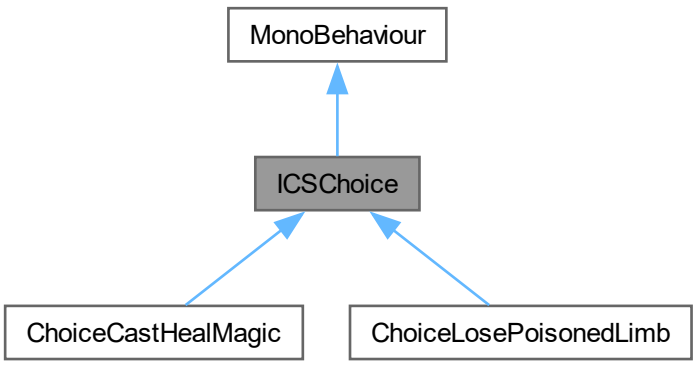
Definition at line 6 of file GameManager.cs.

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

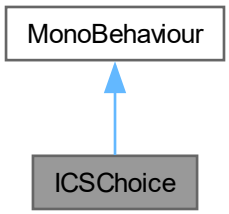
- Managers/GameManager.cs

4.23 ICSChoice Class Reference

Inheritance diagram for ICSChoice:



Collaboration diagram for ICSChoice:



Public Member Functions

- void EventInvoke ()

Events

- static Action ChoicePerformedEvent

4.23.1 Detailed Description

Definition at line 4 of file ICSChoice.cs.

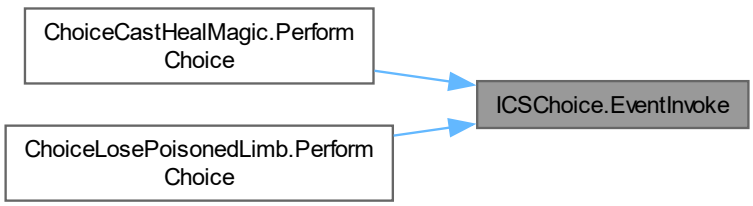
4.23.2 Member Function Documentation

4.23.2.1 EventInvoke()

```
void ICSChoice.EventInvoke ()
```

Definition at line 8 of file ICSChoice.cs.

Here is the caller graph for this function:



4.23.3 Event Documentation

4.23.3.1 ChoicePerformedEvent

```
Action ICSChoice.ChoicePerformedEvent  [static]
```

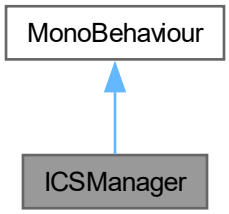
Definition at line 6 of file ICSChoice.cs.

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

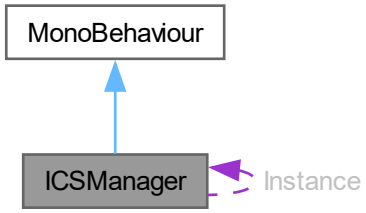
- ICS/ICSChoice.cs

4.24 ICSManager Class Reference

Inheritance diagram for ICSManager:



Collaboration diagram for ICSManager:



Public Member Functions

- void DisplayICSPanel ()
- void HideICSPanel ()

Static Public Attributes

- static ICSManager Instance

4.24.1 Detailed Description

Definition at line 7 of file ICSManager.cs.

4.24.2 Member Function Documentation

4.24.2.1 DisplayICSPanel()

```
void ICSManager.DisplayICSPanel ()
```

Definition at line 25 of file ICSManager.cs.

4.24.2.2 HideICSPanel()

```
void ICSManager.HideICSPanel ()
```

Definition at line 32 of file ICSManager.cs.

4.24.3 Member Data Documentation

4.24.3.1 Instance

```
ICSManager ICSManager.Instance  [static]
```

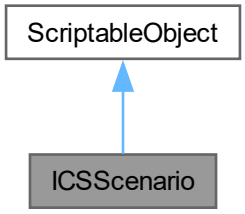
Definition at line 9 of file ICSManager.cs.

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

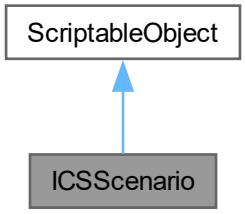
- Managers/ICSManager.cs

4.25 ICSScenario Class Reference

Inheritance diagram for ICSScenario:



Collaboration diagram for ICSScenario:



Public Attributes

- float timer
- string condition
- string[ ] choices

4.25.1 Detailed Description

Definition at line 9 of file ICSScenario.cs.

4.25.2 Member Data Documentation

4.25.2.1 choices

```
string [] ICSScenario.choices
```

Definition at line 16 of file ICSScenario.cs.

4.25.2.2 condition

```
string ICSScenario.condition
```

Definition at line 13 of file ICSScenario.cs.

4.25.2.3 timer

```
float ICSScenario.timer
```

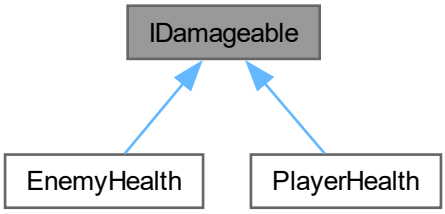
Definition at line 12 of file ICSScenario.cs.

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

- ICS/ICSScenario.cs

4.26 IDamageable Interface Reference

Inheritance diagram for IDamageable:



Public Member Functions

- void TakeDamage (float amount)

4.26.1 Detailed Description

Definition at line 1 of file IDamageable.cs.

4.26.2 Member Function Documentation

4.26.2.1 TakeDamage()

```
void IDamageable.TakeDamage (
            float amount)
```

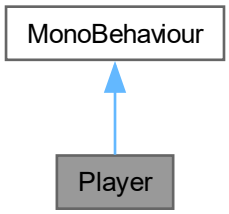
Implemented in EnemyHealth, and PlayerHealth.

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

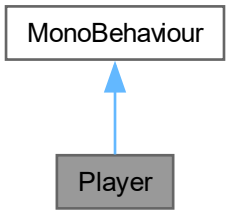
- Extra/IDamageable.cs

4.27 Player Class Reference

Inheritance diagram for Player:



Collaboration diagram for Player:



Public Member Functions

- void ResetPlayer ()

Properties

- PlayerStats Stats [get]

4.27.1 Detailed Description

Definition at line 3 of file Player.cs.

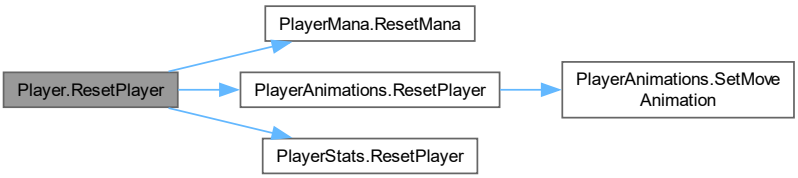
4.27.2 Member Function Documentation

4.27.2.1 ResetPlayer()

```
void Player.ResetPlayer ()
```

Definition at line 19 of file Player.cs.

Here is the call graph for this function:



4.27.3 Property Documentation

4.27.3.1 Stats

```
PlayerStats Player.Stats [get]
```

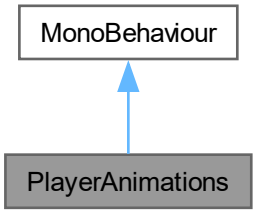
Definition at line 8 of file Player.cs.

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

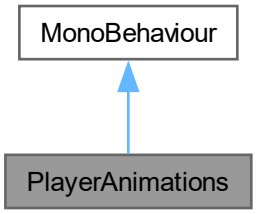
- Player/Player.cs

4.28 PlayerAnimations Class Reference

Inheritance diagram for PlayerAnimations:



Collaboration diagram for PlayerAnimations:



Public Member Functions

- void SetDeadAnimation ()
- void SetMoveBoolTransition (bool value)
- void SetMoveAnimation (Vector2 dir)
- void SetAttackAnimation (bool value)
- void ResetPlayer ()

4.28.1 Detailed Description

Definition at line 3 of file PlayerAnimations.cs.

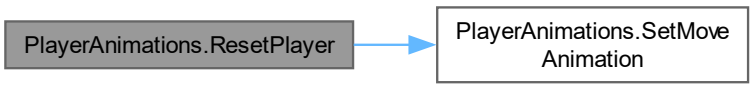
4.28.2 Member Function Documentation

4.28.2.1 ResetPlayer()

```
void PlayerAnimations.ResetPlayer ()
```

Definition at line 40 of file PlayerAnimations.cs.

Here is the call graph for this function:



Here is the caller graph for this function:

4.28.2.2 SetAttackAnimation()

```
void PlayerAnimations.SetAttackAnimation (
            bool value)
```

Definition at line 35 of file PlayerAnimations.cs.

4.28.2.3 SetDeadAnimation()

```
void PlayerAnimations.SetDeadAnimation ()
```

Definition at line 19 of file PlayerAnimations.cs.

4.28.2.4 SetMoveAnimation()

```
void PlayerAnimations.SetMoveAnimation (
            Vector2 dir)
```

Definition at line 29 of file PlayerAnimations.cs.

Here is the caller graph for this function:

4.28.2.5 SetMoveBoolTransition()

```
void PlayerAnimations.SetMoveBoolTransition (
            bool value)
```

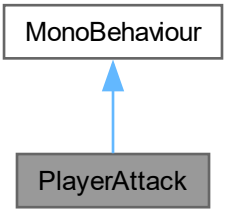
Definition at line 24 of file PlayerAnimations.cs.

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

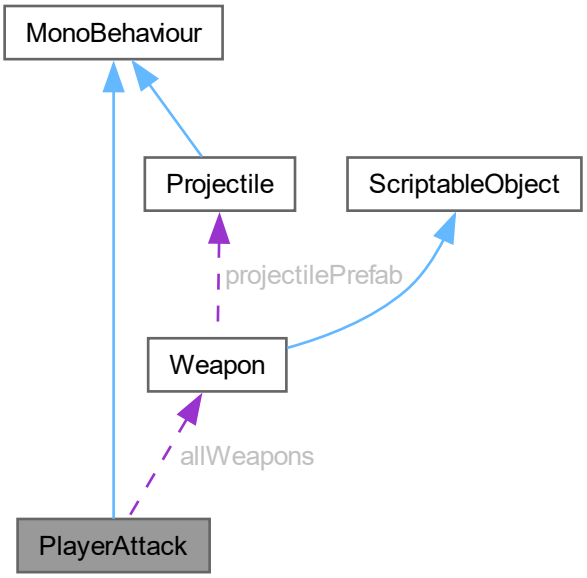
- Player/PlayerAnimations.cs

4.29 PlayerAttack Class Reference

Inheritance diagram for PlayerAttack:



Collaboration diagram for PlayerAttack:



Public Member Functions

- void EquipWeapon (Weapon newWeapon)

Public Attributes

- Weapon[] allWeapons

Properties

- Weapon CurrentWeapon `[get, set]`

4.29.1 Detailed Description

Definition at line 6 of file PlayerAttack.cs.

4.29.2 Member Function Documentation

4.29.2.1 EquipWeapon()

```
void PlayerAttack.EquipWeapon (
            Weapon newWeapon)
```

Definition at line 102 of file PlayerAttack.cs.

4.29.3 Member Data Documentation

4.29.3.1 allWeapons

```
Weapon [] PlayerAttack.allWeapons
```

Definition at line 11 of file PlayerAttack.cs.

4.29.4 Property Documentation

4.29.4.1 CurrentWeapon

```
Weapon PlayerAttack.CurrentWeapon  [get], [set]
```

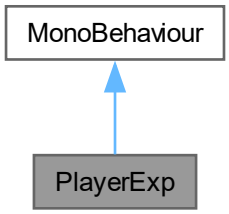
Definition at line 19 of file PlayerAttack.cs.

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

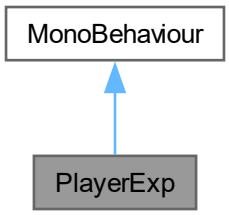
- Player/PlayerAttack.cs

4.30 PlayerExp Class Reference

Inheritance diagram for PlayerExp:



Collaboration diagram for PlayerExp:



Public Member Functions

- void AddExp (float amount)

4.30.1 Detailed Description

Definition at line 3 of file PlayerExp.cs.

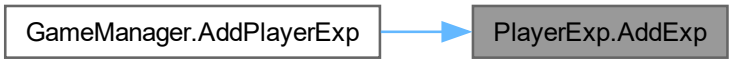
4.30.2 Member Function Documentation

4.30.2.1 AddExp()

```
void PlayerExp.AddExp (
            float amount)
```

Definition at line 16 of file PlayerExp.cs.

Here is the caller graph for this function:

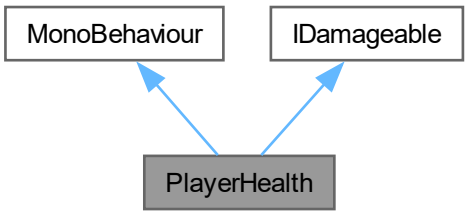


The documentation for this class was generated from the following file:

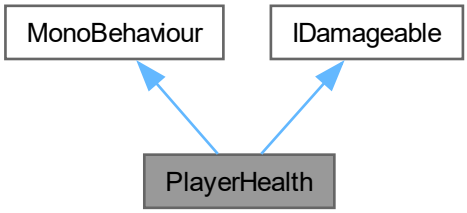
- Player/PlayerExp.cs

4.31 PlayerHealth Class Reference

Inheritance diagram for PlayerHealth:



Collaboration diagram for PlayerHealth:



Public Member Functions

- void TakeDamage (float amount)

Properties

- float CurrentHealth [get]

4.31.1 Detailed Description

Definition at line 3 of file PlayerHealth.cs.

4.31.2 Member Function Documentation

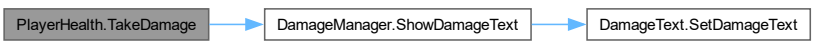
4.31.2.1 TakeDamage()

```
void PlayerHealth.TakeDamage (
            float amount)
```

Implements IDamageable.

Definition at line 24 of file PlayerHealth.cs.

Here is the call graph for this function:



4.31.3 Property Documentation

4.31.3.1 CurrentHealth

```
float PlayerHealth.CurrentHealth  [get]
```

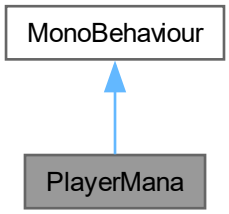
Definition at line 8 of file PlayerHealth.cs.

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

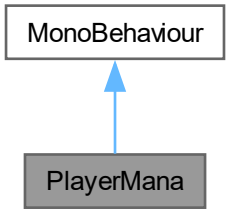
- Player/PlayerHealth.cs

4.32 PlayerMana Class Reference

Inheritance diagram for PlayerMana:



Collaboration diagram for PlayerMana:



Public Member Functions

- void UseMana (float amount)
- void ResetMana ()

Properties

- float CurrentMana [get]

4.32.1 Detailed Description

Definition at line 4 of file PlayerMana.cs.

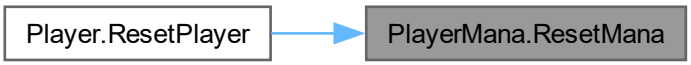
4.32.2 Member Function Documentation

4.32.2.1 ResetMana()

```
void PlayerMana.ResetMana ()
```

Definition at line 26 of file PlayerMana.cs.

Here is the caller graph for this function:



4.32.2.2 UseMana()

```
void PlayerMana.UseMana (
            float amount)
```

Definition at line 20 of file PlayerMana.cs.

Here is the caller graph for this function:

4.32.3 Property Documentation

4.32.3.1 CurrentMana

```
float PlayerMana.CurrentMana [get]
```

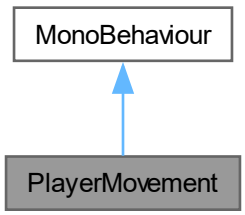
Definition at line 9 of file PlayerMana.cs.

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

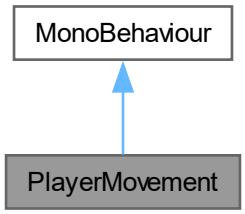
- Player/PlayerMana.cs

4.33 PlayerMovement Class Reference

Inheritance diagram for PlayerMovement:



Collaboration diagram for PlayerMovement:



Properties

- Vector2 MoveDirection [get]

4.33.1 Detailed Description

Definition at line 3 of file PlayerMovement.cs.

4.33.2 Property Documentation

4.33.2.1 MoveDirection

```
Vector2 PlayerMovement.MoveDirection  [get]
```

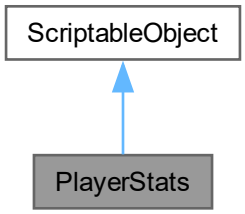
Definition at line 9 of file PlayerMovement.cs.

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

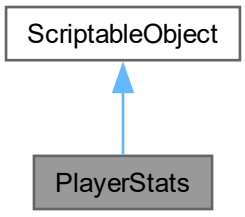
- Player/PlayerMovement.cs

4.34 PlayerStats Class Reference

Inheritance diagram for PlayerStats:



Collaboration diagram for PlayerStats:



Public Member Functions

- void ResetPlayer ()

Public Attributes

- int level
- float speed
- float health
- float maxHealth
- float mana
- float maxMana
- float currentExp
- float nextLevelExp
- float initialNextLevelExp
- float expMultiplier
- float baseDamage
- float criticalChance
- float criticalDamage
- int strength
- int agility
- int intelligence
- int attributePoints
- float totalExp
- float totalDamage

4.34.1 Detailed Description

Definition at line 12 of file PlayerStats.cs.

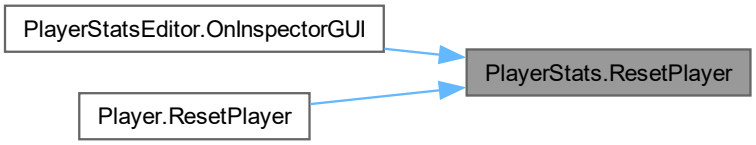
4.34.2 Member Function Documentation

4.34.2.1 ResetPlayer()

```
void PlayerStats.ResetPlayer ()
```

Definition at line 48 of file PlayerStats.cs.

Here is the caller graph for this function:



4.34.3 Member Data Documentation

4.34.3.1 agility

```
int PlayerStats.agility
```

Definition at line 41 of file PlayerStats.cs.

4.34.3.2 attributePoints

```
int PlayerStats.attributePoints
```

Definition at line 43 of file PlayerStats.cs.

4.34.3.3 baseDamage

```
float PlayerStats.baseDamage
```

Definition at line 34 of file PlayerStats.cs.

4.34.3.4 criticalChance

```
float PlayerStats.criticalChance
```

Definition at line 35 of file PlayerStats.cs.

4.34.3.5 criticalDamage

```
float PlayerStats.criticalDamage
```

Definition at line 36 of file PlayerStats.cs.

4.34.3.6 currentExp

```
float PlayerStats.currentExp
```

Definition at line 28 of file PlayerStats.cs.

4.34.3.7 expMultiplier

```
float PlayerStats.expMultiplier
```

Definition at line 31 of file PlayerStats.cs.

4.34.3.8 health

```
float PlayerStats.health
```

Definition at line 20 of file PlayerStats.cs.

4.34.3.9 initialNextLevelExp

```
float PlayerStats.initialNextLevelExp
```

Definition at line 30 of file PlayerStats.cs.

4.34.3.10 intelligence

```
int PlayerStats.intelligence
```

Definition at line 42 of file PlayerStats.cs.

4.34.3.11 level

```
int PlayerStats.level
```

Definition at line 15 of file PlayerStats.cs.

4.34.3.12 mana

```
float PlayerStats.mana
```

Definition at line 24 of file PlayerStats.cs.

4.34.3.13 maxHealth

```
float PlayerStats.maxHealth
```

Definition at line 21 of file PlayerStats.cs.

4.34.3.14 maxMana

```
float PlayerStats.maxMana
```

Definition at line 25 of file PlayerStats.cs.

4.34.3.15 nextLevelExp

```
float PlayerStats.nextLevelExp
```

Definition at line 29 of file PlayerStats.cs.

4.34.3.16 speed

```
float PlayerStats.speed
```

Definition at line 16 of file PlayerStats.cs.

4.34.3.17 strength

```
int PlayerStats.strength
```

Definition at line 39 of file PlayerStats.cs.

4.34.3.18 totalDamage

```
float PlayerStats.totalDamage
```

Definition at line 46 of file PlayerStats.cs.

4.34.3.19 totalExp

```
float PlayerStats.totalExp
```

Definition at line 45 of file PlayerStats.cs.

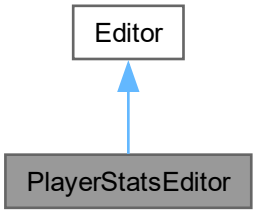
The documentation for this class was generated from the following file:

- Player/PlayerStats.cs

4.35 PlayerStatsEditor Class Reference

Inheritance diagram for PlayerStatsEditor:

Collaboration diagram for PlayerStatsEditor:



Public Member Functions

- override void OnInspectorGUI ()

4.35.1 Detailed Description

Definition at line 6 of file PlayerStatsEditor.cs.

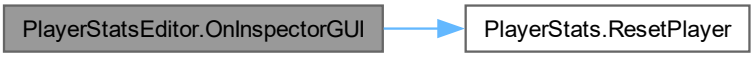
4.35.2 Member Function Documentation

4.35.2.1 OnInspectorGUI()

```
override void PlayerStatsEditor.OnInspectorGUI ()
```

Definition at line 10 of file PlayerStatsEditor.cs.

Here is the call graph for this function:

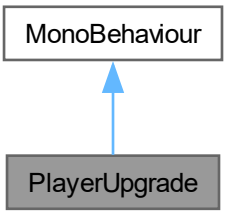


The documentation for this class was generated from the following file:

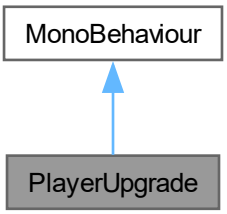
- Player/Editor/PlayerStatsEditor.cs

4.36 PlayerUpgrade Class Reference

Inheritance diagram for PlayerUpgrade:



Collaboration diagram for PlayerUpgrade:



Events

- static Action OnPlayerUpgradeEvent

4.36.1 Detailed Description

Definition at line 7 of file PlayerUpgrade.cs.

4.36.2 Event Documentation

4.36.2.1 OnPlayerUpgradeEvent

```
Action PlayerUpgrade.OnPlayerUpgradeEvent  [static]
```

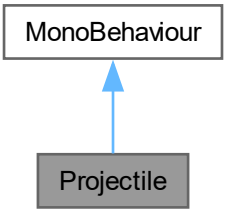
Definition at line 9 of file PlayerUpgrade.cs.

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

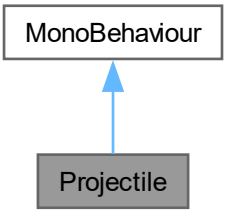
- Player/PlayerUpgrade.cs

4.37 Projectile Class Reference

Inheritance diagram for Projectile:



Collaboration diagram for Projectile:



Properties

- Vector3 Direction `[get, set]`
- float Damage `[get, set]`

4.37.1 Detailed Description

Definition at line 6 of file Projectile.cs.

4.37.2 Property Documentation

4.37.2.1 Damage

```
float Projectile.Damage [get], [set]
```

Definition at line 12 of file Projectile.cs.

4.37.2.2 Direction

```
Vector3 Projectile.Direction  [get], [set]
```

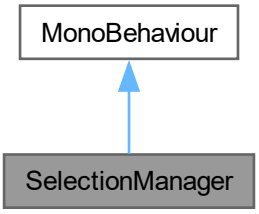
Definition at line 11 of file Projectile.cs.

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

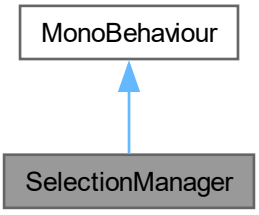
- Weapon/Projectile.cs

4.38 SelectionManager Class Reference

Inheritance diagram for SelectionManager:



Collaboration diagram for SelectionManager:



Events

- static Action< EnemyAI > OnEnemySelectedEvent
- static Action OnNoSelectionEvent

4.38.1 Detailed Description

Definition at line 7 of file SelectionManager.cs.

4.38.2 Event Documentation

4.38.2.1 OnEnemySelectedEvent

```
Action<EnemyAI> SelectionManager.OnEnemySelectedEvent  [static]
```

Definition at line 9 of file SelectionManager.cs.

4.38.2.2 OnNoSelectionEvent

```
Action SelectionManager.OnNoSelectionEvent  [static]
```

Definition at line 10 of file SelectionManager.cs.

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

- Managers/SelectionManager.cs

4.39 SettingsUpgrade Class Reference

Public Attributes

- string name
- float damageUpgrade
- float healthUpgrade
- float manaUpgrade
- float criticalChanceUpgrade
- float criticalDamageUpgrade
- float speedUpgrade

4.39.1 Detailed Description

Definition at line 66 of file PlayerUpgrade.cs.

4.39.2 Member Data Documentation

4.39.2.1 criticalChanceUpgrade

```
float SettingsUpgrade.criticalChanceUpgrade
```

Definition at line 74 of file PlayerUpgrade.cs.

4.39.2.2 criticalDamageUpgrade

```
float SettingsUpgrade.criticalDamageUpgrade
```

Definition at line 75 of file PlayerUpgrade.cs.

4.39.2.3 damageUpgrade

```
float SettingsUpgrade.damageUpgrade
```

Definition at line 71 of file PlayerUpgrade.cs.

4.39.2.4 healthUpgrade

```
float SettingsUpgrade.healthUpgrade
```

Definition at line 72 of file PlayerUpgrade.cs.

4.39.2.5 manaUpgrade

```
float SettingsUpgrade.manaUpgrade
```

Definition at line 73 of file PlayerUpgrade.cs.

4.39.2.6 name

```
string SettingsUpgrade.name
```

Definition at line 68 of file PlayerUpgrade.cs.

4.39.2.7 speedUpgrade

```
float SettingsUpgrade.speedUpgrade
```

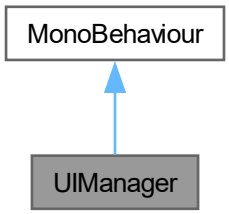
Definition at line 76 of file PlayerUpgrade.cs.

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

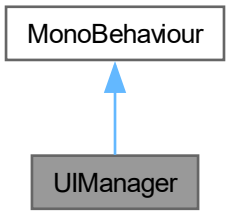
- Player/PlayerUpgrade.cs

4.40 UIManager Class Reference

Inheritance diagram for UIManager:



Collaboration diagram for UIManager:



Public Member Functions

- void DisplayStatsPanel ()

4.40.1 Detailed Description

Definition at line 6 of file UIManager.cs.

4.40.2 Member Function Documentation

4.40.2.1 DisplayStatsPanel()

```
void UIManager.DisplayStatsPanel ()
```

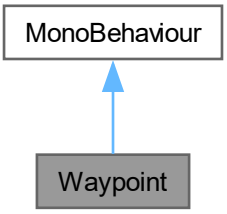
Definition at line 79 of file UIManager.cs.

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

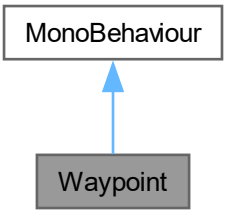
- Managers/UIManager.cs

4.41 Waypoint Class Reference

Inheritance diagram for Waypoint:



Collaboration diagram for Waypoint:



Public Member Functions

- Vector3 GetPosition (int pointIndex)

Properties

- Vector3[] Points `[get]`
- Vector3 EntityPosition `[get, set]`

4.41.1 Detailed Description

Definition at line 6 of file Waypoint.cs.

4.41.2 Member Function Documentation

4.41.2.1 GetPosition()

```
Vector3 Waypoint.GetPosition (
            int pointIndex)
```

Definition at line 22 of file Waypoint.cs.

4.41.3 Property Documentation

4.41.3.1 EntityPosition

```
Vector3 Waypoint.EntityPosition  [get], [set]
```

Definition at line 12 of file Waypoint.cs.

4.41.3.2 Points

```
Vector3 [] Waypoint.Points  [get]
```

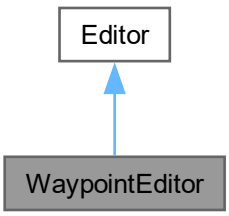
Definition at line 11 of file Waypoint.cs.

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

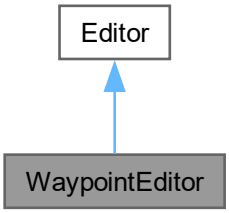
- Waypoint/Waypoint.cs

4.42 WaypointEditor Class Reference

Inheritance diagram for WaypointEditor:



Collaboration diagram for WaypointEditor:



4.42.1 Detailed Description

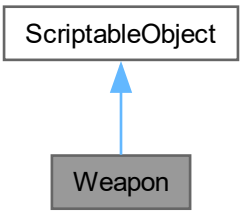
Definition at line 7 of file WaypointEditor.cs.

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

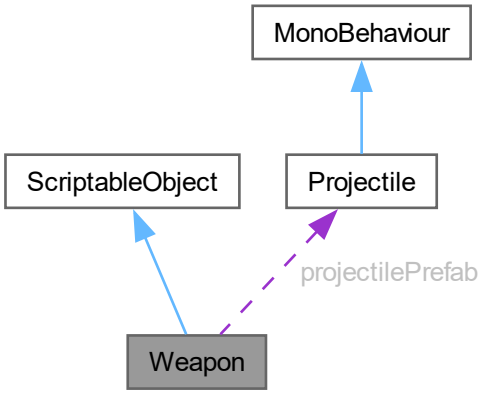
- Waypoint/Editor/WaypointEditor.cs

4.43 Weapon Class Reference

Inheritance diagram for Weapon:



Collaboration diagram for Weapon:



Public Attributes

- Sprite icon
- WeaponType weaponType
- float damage
- Projectile projectilePrefab
- float requiredMana

4.43.1 Detailed Description

Definition at line 13 of file Weapon.cs.

4.43.2 Member Data Documentation

4.43.2.1 damage

```
float Weapon.damage
```

Definition at line 18 of file Weapon.cs.

4.43.2.2 icon

```
Sprite Weapon.icon
```

Definition at line 16 of file Weapon.cs.

4.43.2.3 projectilePrefab

```
Projectile Weapon.projectilePrefab
```

Definition at line 21 of file Weapon.cs.

4.43.2.4 requiredMana

```
float Weapon.requiredMana
```

Definition at line 22 of file Weapon.cs.

4.43.2.5 weaponType

```
WeaponType Weapon.weaponType
```

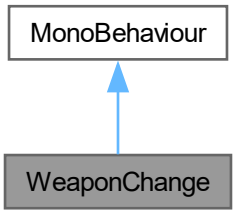
Definition at line 17 of file Weapon.cs.

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

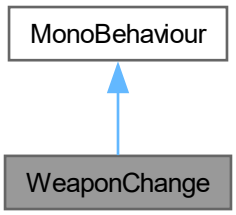
- Weapon/Weapon.cs

4.44 WeaponChange Class Reference

Inheritance diagram for WeaponChange:



Collaboration diagram for WeaponChange:



Public Member Functions

- void ChangeWeapon ()

4.44.1 Detailed Description

Definition at line 8 of file WeaponChange.cs.

4.44.2 Member Function Documentation

4.44.2.1 ChangeWeapon()

```
void WeaponChange.ChangeWeapon ()
```

Definition at line 24 of file WeaponChange.cs.

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

- Weapon/WeaponChange.cs

Chapter 5

File Documentation

5.1 Enemy/EnemyAI.cs File Reference

Classes

- class EnemyAI

5.2 EnemyAI.cs

Go to the documentation of this file.

```
00001 using System;
00002 using UnityEngine;
00003
00004 public class EnemyAI : MonoBehaviour
00005 {
00006     [SerializeField] private string initState;
00007     [SerializeField] private FSMState[] states;
00008
00009     public FSMState CurrentState { get; set; }
00010     public Transform Player { get; set; }
00011
00012     private void Start()
00013     {
00014         ChangeState(initState);
00015     }
00016
00017     private void Update()
00018     {
00019         CurrentState?.UpdateState(this);
00020     }
00021
00022     public void ChangeState(string newStateID)
00023     {
00024         FSMState newState = GetState(newStateID);
00025         if (newState == null) return;
00026         CurrentState = newState;
00027     }
00028
00029     private FSMState GetState(string newStateID)
00030     {
00031         for (int i = 0; i < states.Length; i++)
00032         {
00033             if (states[i].id == newStateID)
00034             {
00035                 return states[i];
00036             }
00037         }
00038
00039         return null;
00040     }
00041 }
```

5.3 Enemy/EnemyHealth.cs File Reference

Classes

- class EnemyHealth

5.4 EnemyHealth.cs

Go to the documentation of this file.

```
00001 using System;
00002 using UnityEngine;
00003
00004 public class EnemyHealth : MonoBehaviour, IDamageable
00005 {
00006     public static event Action OnEnemyDeadEvent;
00007
00008     [Header("Config")]
00009     [SerializeField] private float health;
00010
00011     public float CurrentHealth { get; set; }
00012     private readonly int _dead = Animator.StringToHash("Dead");
00013
00014     private Animator _animator;
00015     private EnemyAI _enemyAI;
00016     private EnemyLoot _enemyLoot;
00017     private EnemySelector _enemySelector;
00018
00019     private void Awake()
00020     {
00021         _enemyLoot = GetComponent<EnemyLoot>();
00022         _animator = GetComponent<Animator>();
00023         _enemyAI = GetComponent<EnemyAI>();
00024         _enemySelector = GetComponent<EnemySelector>();
00025     }
00026
00027     private void Start()
00028     {
00029         CurrentHealth = health;
00030     }
00031
00032     public void TakeDamage(float amount)
00033     {
00034         CurrentHealth -= amount;
00035
00036         if (CurrentHealth <= 0f)
00037         {
00038             DisableEnemy();
00039         }
00040         else
00041         {
00042             DamageManager.Instance.ShowDamageText(amount, transform);
00043         }
00044
00045     }
00046
00047     private void DisableEnemy()
00048     {
00049         _animator.SetTrigger(_dead);
00050         _enemyAI.enabled = false;
00051         _enemySelector.NoSelectionCallBack();
00052         gameObject.layer = LayerMask.NameToLayer("Ignore Raycast");
00053         OnEnemyDeadEvent?.Invoke();
00054         GameManager.Instance.AddPlayerExp(_enemyLoot.ExpDrop);
00055     }
00056 }
```

5.5 Enemy/EnemyLoot.cs File Reference

Classes

- class EnemyLoot

5.6 EnemyLoot.cs

Go to the documentation of this file.

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 public class EnemyLoot : MonoBehaviour
00006 {
00007     [Header("Config")]
00008     [SerializeField] private float expDrop;
00009
00010     public float ExpDrop => expDrop;
00011 }
```

5.7 Enemy/EnemySelector.cs File Reference

Classes

- class EnemySelector

5.8 EnemySelector.cs

Go to the documentation of this file.

```
00001 using System;
00002 using Unity.VisualScripting;
00003 using UnityEngine;
00004
00005 public class EnemySelector : MonoBehaviour
00006 {
00007     [Header("Config")]
00008     [SerializeField] private GameObject selectorSprite;
00009
00010     private EnemyAI _enemyAI;
00011
00012     private void Awake()
00013     {
00014         _enemyAI = GetComponent<EnemyAI>();
00015     }
00016
00017     private void EnemySelectedCallBack(EnemyAI enemySelected)
00018     {
00019         selectorSprite?.SetActive(enemySelected == _enemyAI);
00020     }
00021
00022     public void NoSelectionCallBack()
00023     {
00024         selectorSprite?.SetActive(false);
00025     }
00026
00027     private void OnEnable()
00028     {
00029         SelectionManager.OnEnemySelectedEvent += EnemySelectedCallBack;
00030         SelectionManager.OnNoSelectionEvent += NoSelectionCallBack;
00031     }
00032
00033     private void OnDisable()
00034     {
00035         SelectionManager.OnEnemySelectedEvent -= EnemySelectedCallBack;
00036         SelectionManager.OnNoSelectionEvent += NoSelectionCallBack;
00037     }
00038 }
```

5.9 Enemy/FSM/Actions/ActionAttack.cs File Reference

Classes

- class ActionAttack

5.10 ActionAttack.cs

Go to the documentation of this file.

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using UnityEngine;
00005
00006 public class ActionAttack : FSMAction
00007 {
00008     [Header("Config")]
00009     [SerializeField] private float attackDamage;
00010     [SerializeField] private float attackCooldown;
00011
00012     private EnemyAI _enemyAI;
00013     private float _timer;
00014
00015     private void Awake()
00016     {
00017         _enemyAI = GetComponent<EnemyAI>();
00018     }
00019
00020     public override void Act()
00021     {
00022         AttackPlayer();
00023     }
00024
00025     private void AttackPlayer()
00026     {
00027         if (!_enemyAI.Player) return;
00028         _timer -= Time.deltaTime;
00029         if (_timer > 0f) return;
00030         PlayerHealth player = _enemyAI.Player.GetComponent<PlayerHealth>();
00031         player.TakeDamage(attackDamage);
00032         _timer = attackCooldown;
00033     }
00034
00035 }
```

5.11 Enemy/FSM/Actions/ActionChase.cs File Reference

Classes

- class ActionChase

5.12 ActionChase.cs

Go to the documentation of this file.

```
00001 using UnityEngine;
00002
00003 public class ActionChase : FSMAction
00004 {
00005     [Header("Config")]
00006     [SerializeField] private float chaseSpeed;
00007
00008     private EnemyAI _enemyAI;
00009
00010     private void Awake()
00011     {
00012         _enemyAI = GetComponent<EnemyAI>();
00013     }
00014
00015     public override void Act()
00016     {
00017         ChasePlayer();
00018     }
00019
00020     private void ChasePlayer()
00021     {
00022         if (!_enemyAI.Player) return;
00023         Vector3 directionToPlayer = _enemyAI.Player.position - transform.position;
00024         if (directionToPlayer.magnitude >= 1.3f)
00025         {
00026             transform.Translate(directionToPlayer.normalized * (chaseSpeed * Time.deltaTime));
00027         }
00028     }
00029 }
```

5.13 Enemy/FSM/Actions/ActionPatrol.cs File Reference

Classes

- class ActionPatrol

5.14 ActionPatrol.cs

Go to the documentation of this file.

```
00001 using UnityEngine;
00002
00003
00004 public class ActionPatrol : FSMAction
00005 {
00006     [Header("Config")]
00007     [SerializeField] private float patrolSpeed;
00008
00009     private Waypoint _waypoint;
00010     private int _pointIndex;
00011     private Vector3 _nextPosition;
00012
00013     private void Awake()
00014     {
00015         _waypoint = GetComponent<Waypoint>();
00016     }
00017
00018     public override void Act()
00019     {
00020         FollowPath();
00021     }
00022
00023     private void FollowPath()
00024     {
00025         transform.position = Vector3.MoveTowards(transform.position, GetCurrentPosition(), patrolSpeed
      * Time.deltaTime);
00026         if (Vector3.Distance(transform.position, GetCurrentPosition()) <= 0.1f)
00027         {
00028             UpdateNextPosition();
00029         }
00030     }
00031
00032     private void UpdateNextPosition()
00033     {
00034         _pointIndex++;
00035         if (_pointIndex > _waypoint.Points.Length - 1)
00036         {
00037             _pointIndex = 0;
00038         }
00039     }
00040
00041     private Vector3 GetCurrentPosition()
00042     {
00043         return _waypoint.GetPosition(_pointIndex);
00044     }
00045 }
```

5.15 Enemy/FSM/Actions/ActionWander.cs File Reference

Classes

- class ActionWander

Typedefs

- using Random = UnityEngine.Random

5.15.1 Typedef Documentation

5.15.1.1 Random

```
using Random = UnityEngine.Random
```

Definition at line 2 of file ActionWander.cs.

5.16 ActionWander.cs

Go to the documentation of this file.

```
00001 using UnityEngine;
00002 using Random = UnityEngine.Random;
00003
00004 public class ActionWander : FSMAction
00005 {
00006     [Header("Config")]
00007     [SerializeField] private float wanderSpeed;
00008     [SerializeField] private float wanderTime;
00009     [SerializeField] private Vector2 moveRange;
00010
00011     private void Start()
00012     {
00013         GetNewDestination();
00014     }
00015
00016     private Vector3 _movePosition;
00017     private float _timer;
00018     public override void Act()
00019     {
00020         _timer -= Time.deltaTime;
00021
00022         Vector3 moveDirection = (_movePosition - transform.position).normalized;
00023         Vector3 movement = moveDirection * (wanderSpeed * Time.deltaTime);
00024
00025         if (Vector3.Distance(transform.position, _movePosition) >= 0.5f)
00026         {
00027             transform.Translate(movement);
00028         }
00029
00030         if (!(_timer <= 0f)) return;
00031         GetNewDestination();
00032         _timer = wanderTime;
00033     }
00034
00035     private void GetNewDestination()
00036     {
00037         float randomX = Random.Range(-moveRange.x, moveRange.x);
00038         float randomY = Random.Range(-moveRange.y, moveRange.y);
00039
00040         _movePosition = transform.position + new Vector3(randomX, randomY);
00041     }
00042
00043     private void OnDrawGizmosSelected()
00044     {
00045         if (moveRange == Vector2.zero) return;
00046         Gizmos.color = Color.cyan;
00047         Gizmos.DrawWireCube(transform.position, moveRange * 2f);
00048         Gizmos.DrawLine(transform.position, _movePosition);
00049     }
00050 }
```

5.17 Enemy/FSM/Decisions/DecisionAttackPlayer.cs File Reference

Classes

- class DecisionAttackPlayer

5.18 DecisionAttackPlayer.cs

Go to the documentation of this file.

```
00001 using UnityEngine;
00002 using UnityEngine.Serialization;
00003
00004 public class DecisionAttackPlayer : FSMDecision
00005 {
00006     [FormerlySerializedAs("range")]
00007     [Header("Config")]
00008     [SerializeField] private float attackRange;
00009     [SerializeField] private LayerMask playerMask;
00010
00011     private EnemyAI _enemy;
00012
00013     private void Awake()
00014     {
00015         _enemy = GetComponent<EnemyAI>();
00016     }
00017
00018     public override bool Decide()
00019     {
00020         return PlayerInAttackRange();
00021     }
00022
00023     private bool PlayerInAttackRange()
00024     {
00025         if (!_enemy.Player) return false;
00026         Collider2D playerCollider = Physics2D.OverlapCircle(_enemy.transform.position, attackRange,
      playerMask);
00027         return playerCollider;
00028     }
00029
00030     private void OnDrawGizmosSelected()
00031     {
00032         Gizmos.color = Color.yellow;
00033         Gizmos.DrawWireSphere(transform.position, attackRange);
00034     }
00035 }
```

5.19 Enemy/FSM/Decisions/DecisionDetectPlayer.cs File Reference

Classes

- class DecisionDetectPlayer

5.20 DecisionDetectPlayer.cs

Go to the documentation of this file.

```
00001 using UnityEngine;
00002 using UnityEngine.Serialization;
00003
00004
00005 public class DecisionDetectPlayer : FSMDecision
00006 {
00007     [Header("Config")]
00008     [SerializeField] private float detectRange;
00009     [SerializeField] private LayerMask playerMask;
00010
00011     private EnemyAI _enemy;
00012
00013     private void Awake()
00014     {
00015         _enemy = GetComponent<EnemyAI>();
00016     }
00017
00018     public override bool Decide()
00019     {
00020         return DetectPlayer();
00021     }
00022
00023     private bool DetectPlayer()
```

```
00024     {
00025         Collider2D playerCollider = Physics2D.OverlapCircle(_enemy.transform.position, detectRange,
    playerMask);
00026         if (playerCollider)
00027         {
00028             _enemy.Player = playerCollider.transform;
00029             return true;
00030         }
00031 
00032         _enemy.Player = null;
00033         return false;
00034     }
00035 
00036     private void OnDrawGizmosSelected()
00037     {
00038         Gizmos.color = Color.red;
00039         Gizmos.DrawWireSphere(transform.position, detectRange);
00040     }
00041 }
```

5.21 Enemy/FSM/FSMAction.cs File Reference

Classes

- class FSMAction

5.22 FSMAction.cs

Go to the documentation of this file.

```
00001 using UnityEngine;
00002 
00003 public abstract class FSMAction : MonoBehaviour
00004 {
00005     public abstract void Act();
00006 }
```

5.23 Enemy/FSM/FSMDecision.cs File Reference

Classes

- class FSMDecision

5.24 FSMDecision.cs

Go to the documentation of this file.

```
00001 using UnityEngine;
00002 
00003 public abstract class FSMDecision : MonoBehaviour
00004 {
00005         public abstract bool Decide();
00006 }
```

5.25 Enemy/FSM/FSMState.cs File Reference

Classes

- class FSMState

5.26 FSMState.cs

Go to the documentation of this file.

```
00001 using System;
00002
00003 [Serializable]
00004 public class FSMState
00005 {
00006         public string id;
00007         public FSMAction[] actions;
00008         public FSMTransition[] transitions;
00009
00010         public void UpdateState(EnemyAI enemyAI)
00011         {
00012                 ExecuteActions();
00013                 ExecuteTransitions(enemyAI);
00014         }
00015
00016         private void ExecuteActions()
00017         {
00018                 foreach (var action in actions)
00019                 {
00020                         action.Act();
00021                 }
00022         }
00023
00024         private void ExecuteTransitions(EnemyAI enemyAI)
00025         {
00026                 if (transitions is not { Length: > 0 }) return;
00027                 foreach (var transition in transitions)
00028                 {
00029                         bool value = transition.decision.Decide();
00030                         enemyAI.ChangeState(value ? transition.trueState : transition.falseState);
00031                 }
00032         }
00033 }
```

5.27 Enemy/FSM/FSMTransition.cs File Reference

Classes

- class FSMTransition

5.28 FSMTransition.cs

Go to the documentation of this file.

```
00001 using System;
00002 using UnityEngine.Serialization;
00003
00004 [Serializable]
00005 public class FSMTransition
00006 {
00007     [FormerlySerializedAs("Decision")] public FSMDecision decision;
00008     [FormerlySerializedAs("True State")] public string trueState;
00009     [FormerlySerializedAs("False State")] public string falseState;
00010 }
```

5.29 Extra/AttributeButton.cs File Reference

Classes

- class AttributeButton

5.30 AttributeButton.cs

Go to the documentation of this file.

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using UnityEngine;
00005
00006 public class AttributeButton : MonoBehaviour
00007 {
00008     public static event Action<AttributeType> OnAttributeSelectedEvent;
00009
00010     [Header("Config")]
00011     [SerializeField] private AttributeType attribute;
00012
00013     public void SelectAttribute()
00014     {
00015         OnAttributeSelectedEvent?.Invoke(attribute);
00016     }
00017 }
```

5.31 Extra/ExitButton.cs File Reference

Classes

- class ExitButton

5.32 ExitButton.cs

Go to the documentation of this file.

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 using UnityEngine.SceneManagement;
00005
00006 public class ExitButton : MonoBehaviour
00007 {
00008     public void ExitGame()
00009     {
00010         SceneManager.LoadScene(0);
00011     }
00012 }
```

5.33 Extra/IDamageable.cs File Reference

Classes

- interface IDamageable

5.34 IDamageable.cs

Go to the documentation of this file.

```
00001 public interface IDamageable
00002 {
00003     void TakeDamage(float amount);
00004
00005 }
```

5.35 ICS/ICSChoice.cs File Reference

Classes

- class ICSChoice

5.36 ICSChoice.cs

Go to the documentation of this file.

```
00001 using System;
00002 using UnityEngine;
00003
00004 public class ICSChoice : MonoBehaviour
00005 {
00006     public static event Action ChoicePerformedEvent;
00007
00008     public void EventInvoke()
00009     {
00010         ChoicePerformedEvent?.Invoke();
00011     }
00012 }
```

5.37 ICS/ICSScenario.cs File Reference

Classes

- class ICSScenario

5.38 ICSScenario.cs

Go to the documentation of this file.

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using TMPro;
00005 using UnityEngine;
00006 using UnityEngine.Serialization;
00007
00008 [CreateAssetMenu(menuName = "ICS Scenario")]
00009 public class ICSScenario : ScriptableObject
00010 {
00011     [Header("Config")]
00012     public float timer;
00013     [TextArea] public string condition;
00014
00015     [Header("Choices")]
00016     [TextArea] public string[] choices;
00017 }
```

5.39 ICS/Poisoned/ChoiceCastHealMagic.cs File Reference

Classes

- class ChoiceCastHealMagic

5.40 ChoiceCastHealMagic.cs

Go to the documentation of this file.

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using Unity.Collections;
00004 using Unity.VisualScripting;
00005 using UnityEngine;
00006
00007 public class ChoiceCastHealMagic : ICSChoice
00008 {
00009     [SerializeField] private Player player;
00010     [SerializeField] private float manaCost;
00011
00012     public void PerformChoice()
00013     {
00014         var mana = player.GetComponent<PlayerMana>();
00015
00016         if (mana.CurrentMana >= manaCost)
00017         {
00018             mana.UseMana(manaCost);
00019         }
00020         else
00021         {
00022             player.GetComponent<PlayerHealth>().TakeDamage(manaCost);
00023         }
00024         EventInvoke();
00025     }
00026 }
```

5.41 ICS/Poisoned/ChoiceLosePoisonedLimb.cs File Reference

Classes

- class ChoiceLosePoisonedLimb

5.42 ChoiceLosePoisonedLimb.cs

Go to the documentation of this file.

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004
00005 public class ChoiceLosePoisonedLimb : ICSChoice
00006 {
00007     [SerializeField] private Player player;
00008     [SerializeField] private float limbCost;
00009
00010     public void PerformChoice()
00011     {
00012         player.GetComponent<PlayerHealth>().TakeDamage(limbCost);
00013         EventInvoke();
00014     }
00015 }
```

5.43 ICS/Poisoned/ConditionPoisoned.cs File Reference

Classes

- class ConditionPoisoned

Typedefs

- using Random = UnityEngine.Random

5.43.1 Typedef Documentation

5.43.1.1 Random

```
using Random = UnityEngine.Random
```

Definition at line 6 of file ConditionPoisoned.cs.

5.44 ConditionPoisoned.cs

Go to the documentation of this file.

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using Unity.VisualScripting;
00005 using UnityEngine;
00006 using Random = UnityEngine.Random;
00007
00008 public class ConditionPoisoned : MonoBehaviour
00009 {
00010     public static event Action PoisonConditionEvent;
00011     private EnemyAI _enemyAI;
00012     private FSMState currenState;
00013     private float playerHealth;
00014     private bool isInvoked;
00015     private bool wasInvoked;
00016     private void Awake()
00017     {
00018         _enemyAI = GetComponent<EnemyAI>();
00019         isInvoked = false;
00020         wasInvoked = false;
00021     }
00022
00023     private void Update()
00024     {
00025         if (_enemyAI.Player)
00026         {
00027             currenState = _enemyAI.CurrentState;
00028             playerHealth = _enemyAI.Player.GetComponent<PlayerHealth>().CurrentHealth;
00029             if (currenState.id == "Attack" && playerHealth < 10f)
00030             {
00031                 isInvoked = true;
00032             }
00033         }
00034
00035         if (isInvoked && !wasInvoked)
00036         {
00037             PoisonConditionEvent?.Invoke();
00038             wasInvoked = true;
00039         }
00040     }
00041
00042 }
```

5.45 Managers/DamageManager.cs File Reference

Classes

- class DamageManager

5.46 DamageManager.cs

Go to the documentation of this file.

```
00001 using System;
00002 using UnityEngine;
00003
00004 public class DamageManager : MonoBehaviour
00005 {
00006         public static DamageManager Instance;
00007
00008         [Header("Config")]
00009         [SerializeField] private DamageText damageTextPrefab;
00010
00011         private void Awake()
00012         {
00013                 Instance = this;
00014         }
00015
00016         public void ShowDamageText(float damage, Transform parent)
00017         {
00018                 DamageText damageText = Instantiate(damageTextPrefab, parent);
00019                 damageText.transform.position += Vector3.right * 0.5f;
00020                 damageText.SetDamageText(damage);
00021         }
00022 }
```

5.47 Managers/GameManager.cs File Reference

Classes

- class GameManager

5.48 GameManager.cs

Go to the documentation of this file.

```
00001 using System;
00002 using UnityEngine;
00003
00004 public class GameManager : MonoBehaviour
00005 {
00006     public static GameManager Instance;
00007
00008     [SerializeField] private Player player;
00009
00010     private void Awake()
00011     {
00012         Instance = this;
00013     }
00014
00015     private void Update()
00016     {
00017         if (Input.GetKeyDown(KeyCode.R))
00018         {
00019             player.ResetPlayer();
00020         }
00021     }
00022
00023     public void AddPlayerExp(float expAmount)
00024     {
00025         PlayerExp playerExp = player.GetComponent<PlayerExp>();
00026         playerExp.AddExp(expAmount);
00027     }
00028 }
```

5.49 Managers/ICSManager.cs File Reference

Classes

- class ICSManager

5.50 ICSManager.cs

Go to the documentation of this file.

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using TMPro;
00005 using UnityEngine;
00006
00007 public class ICSManager : MonoBehaviour
00008 {
00009     public static ICSManager Instance;
00010
00011     [Header("Scenarios")] [SerializeField] private ICSScenario scenario;
00012
00013     [Header("ICS Panel")]
00014     [SerializeField] private GameObject ICSPanel;
00015     [SerializeField] private TextMeshProUGUI conditionTMP;
00016     [SerializeField] private TextMeshProUGUI choice1TMP;
00017     [SerializeField] private TextMeshProUGUI choice2TMP;
00018
00019     private void Awake()
00020     {
00021         Instance = this;
00022     }
00023
00024
00025     public void DisplayICSPanel()
00026     {
00027         Time.timeScale = 0f;
00028         UpdateICSPanel();
00029         ICSPanel.SetActive(true);
00030     }
00031
00032     public void HideICSPanel()
00033     {
00034         Time.timeScale = 1f;
00035         ICSPanel.SetActive(false);
00036     }
00037
00038     private void UpdateICSPanel()
00039     {
00040         conditionTMP.text = scenario.condition;
00041         choice1TMP.text = scenario.choices[0];
00042         choice2TMP.text = scenario.choices[1];
00043     }
00044
00045     private void UpgradeCallBack()
00046     {
00047         DisplayICSPanel();
00048     }
00049
00050     private void ChoicePerformedCallBack()
00051     {
00052         HideICSPanel();
00053     }
00054
00055     private void OnEnable()
00056     {
00057         ConditionPoisoned.PoisonConditionEvent += UpgradeCallBack;
00058         ICSChoice.ChoicePerformedEvent += ChoicePerformedCallBack;
00059     }
00060
00061     private void OnDisable()
00062     {
00063         ConditionPoisoned.PoisonConditionEvent -= UpgradeCallBack;
00064         ICSChoice.ChoicePerformedEvent += ChoicePerformedCallBack;
00065     }
00066 }
```

5.51 Managers/SelectionManager.cs File Reference

Classes

- class SelectionManager

5.52 SelectionManager.cs

Go to the documentation of this file.

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using Unity.VisualScripting;
00005 using UnityEngine;
00006
00007 public class SelectionManager : MonoBehaviour
00008 {
00009     public static event Action<EnemyAI> OnEnemySelectedEvent;
00010     public static event Action OnNoSelectionEvent;
00011
00012     [Header("Config")]
00013     [SerializeField] private LayerMask enemyMask;
00014
00015     private Camera _mainCamera;
00016
00017     private void Awake()
00018     {
00019         _mainCamera = Camera.main;
00020     }
00021
00022     private void Update()
00023     {
00024         SelectEnemy();
00025     }
00026
00027     private void SelectEnemy()
00028     {
00029         if (Input.GetMouseButtonDown(0))
00030         {
00031             RaycastHit2D hit = Physics2D.Raycast(_mainCamera.ScreenToWorldPoint(Input.mousePosition),
00032                 Vector2.zero, Mathf.Infinity, enemyMask);
00033
00034             if (hit.collider)
00035             {
00036                 EnemyAI enemyAI = hit.collider.GetComponent<EnemyAI>();
00037                 if(!enemyAI) return;
00038                 EnemyHealth enemyHealth = enemyAI.GetComponent<EnemyHealth>();
00039                 if(enemyHealth.CurrentHealth <= 0f) return;
00040                 OnEnemySelectedEvent?.Invoke(enemyAI);
00041
00042             }
00043             else
00044             {
00045                 OnNoSelectionEvent?.Invoke();
00046             }
00047         }
00048     }
00049 }
```

5.53 Managers/UIManager.cs File Reference

Classes

- class UIManager

5.54 UIManager.cs

Go to the documentation of this file.

```
00001 using System;
00002 using TMPro;
00003 using UnityEngine;
00004 using UnityEngine.UI;
00005
00006 public class UIManager : MonoBehaviour
00007 {
00008     [Header("Player Attack")]
00009     [SerializeField] private Player player;
00010
```

```
00011    [Header("Stats")]
00012    [SerializeField] private PlayerStats stats;
00013
00014    [Header("Bars")]
00015    [SerializeField] private Image healthBar;
00016    [SerializeField] private Image manaBar;
00017    [SerializeField] private Image expBar;
00018
00019    [Header("Text")]
00020    [SerializeField] private TextMeshProUGUI levelTMP;
00021    [SerializeField] private TextMeshProUGUI healthTMP;
00022    [SerializeField] private TextMeshProUGUI manaTMP;
00023    [SerializeField] private TextMeshProUGUI expTMP;
00024
00025    [Header("Stats Panel")]
00026    [SerializeField] private GameObject statsPanel;
00027    [SerializeField] private TextMeshProUGUI statLevelTMP;
00028    [SerializeField] private TextMeshProUGUI statHealthPointsTMP;
00029    [SerializeField] private TextMeshProUGUI statManaPointsTMP;
00030    [SerializeField] private TextMeshProUGUI statExpTMP;
00031    [SerializeField] private TextMeshProUGUI statDamageTMP;
00032    [SerializeField] private TextMeshProUGUI statCriticalDamageTMP;
00033    [SerializeField] private TextMeshProUGUI statCriticalChanceTMP;
00034    [SerializeField] private TextMeshProUGUI statSpeedTMP;
00035    [SerializeField] private TextMeshProUGUI attributePointsTMP;
00036    [SerializeField] private TextMeshProUGUI strengthTMP;
00037    [SerializeField] private TextMeshProUGUI agilityTMP;
00038    [SerializeField] private TextMeshProUGUI intelligenceTMP;
00039
00040    [Header("Weapon")]
00041    [SerializeField] private Image weaponIcon;
00042
00043    [Header("Exit Button")]
00044    [SerializeField] private GameObject exit;
00045
00046    private PlayerAttack _playerAttack;
00047
00048    private void Awake()
00049    {
00050       _playerAttack = player.GetComponent<PlayerAttack>();
00051    }
00052
00053    private void Update()
00054    {
00055       UpdatePlayerHUD();
00056       if (Input.GetKeyDown(KeyCode.M))
00057       {
00058          DisplayStatsPanel();
00059       }
00060    }
00061
00062    private void DisplayExitButton()
00063    {
00064       exit.SetActive(!exit.activeSelf);
00065    }
00066
00067    private void UpdatePlayerHUD()
00068    {
00069       healthBar.fillAmount = Mathf.Lerp(healthBar.fillAmount, stats.health / stats.maxHealth, 10f *
   Time.deltaTime);
00070       manaBar.fillAmount = Mathf.Lerp(manaBar.fillAmount, stats.mana / stats.maxMana, 10f *
   Time.deltaTime);
00071       expBar.fillAmount = Mathf.Lerp(expBar.fillAmount, stats.currentExp / stats.nextLevelExp, 10f *
   Time.deltaTime);
00072
00073       levelTMP.text = $"Level {stats.level}";
00074       healthTMP.text = $"{stats.health} / {stats.maxHealth}";
00075       manaTMP.text = $"{stats.mana} / {stats.maxMana}";
00076       expTMP.text = $"{stats.currentExp} / {stats.nextLevelExp}";
00077    }
00078
00079    public void DisplayStatsPanel()
00080    {
00081          statsPanel.SetActive(!statsPanel.activeSelf);
00082          if (statsPanel.activeSelf)
00083          {
00084             UpdateStatsPanel();
00085          }
00086    }
00087
00088    private void UpdateStatsPanel()
00089    {
00090       statLevelTMP.text = stats.level.ToString();
00091       statHealthPointsTMP.text = $"{stats.health}/{stats.maxHealth}";
00092       statManaPointsTMP.text = $"{stats.mana}/{stats.maxMana}";
00093       statExpTMP.text = stats.totalExp.ToString();
00094       statDamageTMP.text = stats.totalDamage.ToString();
```

```
00095         statCriticalDamageTMP.text = stats.criticalDamage.ToString();
00096         statCriticalChanceTMP.text = stats.criticalChance.ToString();
00097         statSpeedTMP.text = stats.speed.ToString();
00098 
00099         attributePointsTMP.text = $"Points: {stats.attributePoints}";
00100         strengthTMP.text = stats.strength.ToString();
00101         agilityTMP.text = stats.agility.ToString();
00102         intelligenceTMP.text = stats.intelligence.ToString();
00103     }
00104 
00105     private void UpgradeCallBack()
00106     {
00107         UpdateStatsPanel();
00108     }
00109 
00110     private void OnEnable()
00111     {
00112         PlayerUpgrade.OnPlayerUpgradeEvent += UpgradeCallBack;
00113     }
00114 
00115     private void OnDisable()
00116     {
00117         PlayerUpgrade.OnPlayerUpgradeEvent -= UpgradeCallBack;
00118     }
00119 }
```

5.55 Player/Editor/PlayerStatsEditor.cs File Reference

Classes

- class PlayerStatsEditor

5.56 PlayerStatsEditor.cs

Go to the documentation of this file.

```
00001 using UnityEditor;
00002 using UnityEngine;
00003 
00004 [CustomEditor(typeof(PlayerStats))]
00005 
00006 public class PlayerStatsEditor : Editor
00007 {
00008     private PlayerStats StatsTarget => target as PlayerStats;
00009 
00010     public override void OnInspectorGUI()
00011     {
00012         base.OnInspectorGUI();
00013         if (GUILayout.Button("Reset Player"))
00014         {
00015             StatsTarget.ResetPlayer();
00016         }
00017     }
00018 }
```

5.57 Player/Player.cs File Reference

Classes

- class Player

5.58 Player.cs

Go to the documentation of this file.

```
00001 using System;
00002 using UnityEngine;
00003 public class Player : MonoBehaviour
00004 {
00005     [Header("Config")]
00006     [SerializeField] private PlayerStats stats;
00007
00008     public PlayerStats Stats => stats;
00009     private PlayerMana PlayerMana { get; set; }
00010
00011     private PlayerAnimations _animations;
00012
00013     private void Awake()
00014     {
00015         PlayerMana = GetComponent<PlayerMana>();
00016         _animations = GetComponent<PlayerAnimations>();
00017     }
00018
00019     public void ResetPlayer()
00020     {
00021         stats.ResetPlayer();
00022         _animations.ResetPlayer();
00023         PlayerMana.ResetMana();
00024     }
00025 }
```

5.59 Player/PlayerAnimations.cs File Reference

Classes

- class PlayerAnimations

5.60 PlayerAnimations.cs

Go to the documentation of this file.

```
00001 using UnityEngine;
00002
00003 public class PlayerAnimations : MonoBehaviour
00004 {
00005     private readonly int _moveX = Animator.StringToHash("MoveX");
00006     private readonly int _moveY = Animator.StringToHash("MoveY");
00007     private readonly int _moving = Animator.StringToHash("Moving");
00008     private readonly int _dead = Animator.StringToHash("Dead");
00009     private readonly int _revive = Animator.StringToHash("Revive");
00010     private readonly int _attacking = Animator.StringToHash("Attacking");
00011
00012     private Animator _animator;
00013
00014     private void Awake()
00015     {
00016         _animator = GetComponent<Animator>();
00017     }
00018
00019     public void SetDeadAnimation()
00020     {
00021         _animator.SetTrigger(_dead);
00022     }
00023
00024     public void SetMoveBoolTransition(bool value)
00025     {
00026         _animator.SetBool(_moving, value);
00027     }
00028
00029     public void SetMoveAnimation(Vector2 dir)
00030     {
00031         _animator.SetFloat(_moveX, dir.x);
00032         _animator.SetFloat(_moveY, dir.y);
00033     }
00034
```

```
00035     public void SetAttackAnimation(bool value)
00036     {
00037         _animator.SetBool(_attacking, value);
00038     }
00039
00040     public void ResetPlayer()
00041     {
00042         SetMoveAnimation(Vector2.down);
00043         _animator.SetTrigger(_revive);
00044     }
00045 }
```

5.61 Player/PlayerAttack.cs File Reference

Classes

- class PlayerAttack

Typedefs

- using Random = UnityEngine.Random

5.61.1 Typedef Documentation

5.61.1.1 Random

```
using Random = UnityEngine.Random
```

Definition at line 4 of file PlayerAttack.cs.

5.62 PlayerAttack.cs

Go to the documentation of this file.

```
00001 using System;
00002 using System.Collections;
00003 using UnityEngine;
00004 using Random = UnityEngine.Random;
00005
00006 public class PlayerAttack: MonoBehaviour
00007 {
00008     [Header("Config")]
00009     [SerializeField] private PlayerStats stats;
00010     [SerializeField] private Weapon initialWeapon;
00011     [SerializeField] public Weapon[] allWeapons;
00012     [SerializeField] private Transform[] attackPositions;
00013
00014     [Header("Melee Config")]
00015     [SerializeField] private ParticleSystem slashFX;
00016
00017     [SerializeField] private float minDistanceMeleeAttack;
00018
00019     public Weapon CurrentWeapon { get; set; }
00020
00021     private PlayerActions _actions;
00022     private PlayerAnimations _playerAnimations;
00023     private PlayerMovement _playerMovement;
00024     private PlayerMana _playerMana;
00025     private EnemyAI _enemyTarget;
00026     private Coroutine _attackCoroutine;
00027
00028     private Transform _currentAttackPosition;
00029     private float _currentAttackRotation;
00030
```

```
00031
00032
00033     private void Awake()
00034     {
00035         _actions = new PlayerActions();
00036         _playerMana = GetComponent<PlayerMana>();
00037         _playerMovement = GetComponent<PlayerMovement>();
00038         _playerAnimations = GetComponent<PlayerAnimations>();
00039     }
00040
00041     private void Start()
00042     {
00043         EquipWeapon(initialWeapon);
00044     }
00045
00046     private void Update()
00047     {
00048         _actions.Attack.ClickAttack.performed += ctx => Attack();
00049         GetFirePosition();
00050     }
00051
00052     private void Attack()
00053     {
00054         if (!_enemyTarget) return;
00055         if (_attackCoroutine != null)
00056         {
00057             StopCoroutine(_attackCoroutine);
00058         }
00059
00060         _attackCoroutine = StartCoroutine(IEAttack());
00061     }
00062
00063     private IEnumerator IEAttack()
00064     {
00065         if (!_currentAttackPosition) yield break;
00066         if (CurrentWeapon.weaponType == WeaponType.Magic)
00067         {
00068             if (_playerMana.CurrentMana < CurrentWeapon.requiredMana) yield break;
00069             MagicAttack();
00070         }
00071         else
00072         {
00073             MeleeAttack();
00074         }
00075
00076         _playerAnimations.SetAttackAnimation(true);
00077         yield return new WaitForSeconds(0.5f);
00078         _playerAnimations.SetAttackAnimation(false);
00079     }
00080
00081     private void MagicAttack()
00082     {
00083         Quaternion rotation = Quaternion.Euler(new Vector3(0f, 0f, _currentAttackRotation));
00084         Projectile projectile = Instantiate(CurrentWeapon.projectilePrefab,
    _currentAttackPosition.position, rotation);
00085         projectile.Direction = Vector3.up;
00086         projectile.Damage = GetAttackDamage();
00087         _playerMana.UseMana(CurrentWeapon.requiredMana);
00088     }
00089
00090     private void MeleeAttack()
00091     {
00092         slashFX.transform.position = _currentAttackPosition.position;
00093         slashFX.Play();
00094         float currenDistanceToEnemy = Vector3.Distance(_enemyTarget.transform.position,
    transform.position);
00095
00096         if (currenDistanceToEnemy <= minDistanceMeleeAttack)
00097         {
00098             _enemyTarget.GetComponent<IDamageable>().TakeDamage(GetAttackDamage());
00099         }
00100     }
00101
00102     public void EquipWeapon(Weapon newWeapon)
00103     {
00104         CurrentWeapon = newWeapon;
00105         stats.totalDamage = stats.baseDamage + CurrentWeapon.damage;
00106     }
00107
00108     private float GetAttackDamage()
00109     {
00110         float attackDamage = stats.baseDamage;
00111         attackDamage += CurrentWeapon.damage;
00112         float randomPercentage = Random.Range(0f, 100);
00113         if (randomPercentage <= stats.criticalChance)
00114         {
00115             attackDamage += attackDamage * (stats.criticalDamage / 100f);
```

```
00116         }
00117
00118         return attackDamage;
00119     }
00120
00121     private void GetFirePosition()
00122     {
00123         Vector2 moveDirection = _playerMovement.MoveDirection;
00124         switch (moveDirection.x)
00125         {
00126             case > 0f:
00127                 _currentAttackPosition = attackPositions[1];
00128                 _currentAttackRotation = -90f;
00129                 break;
00130             case < 0f:
00131                 _currentAttackPosition = attackPositions[3];
00132                 _currentAttackRotation = 90f;
00133                 break;
00134         }
00135         switch (moveDirection.y)
00136         {
00137             case > 0f:
00138                 _currentAttackPosition = attackPositions[0];
00139                 _currentAttackRotation = 0f;
00140                 break;
00141             case < 0f:
00142                 _currentAttackPosition = attackPositions[2];
00143                 _currentAttackRotation = 180f;
00144                 break;
00145         }
00146     }
00147
00148     private void EnemySelectedCallBack(EnemyAI enemySelected)
00149     {
00150         _enemyTarget = enemySelected;
00151     }
00152
00153     private void NoSelectionCallBack()
00154     {
00155         _enemyTarget = null;
00156     }
00157
00158
00159     private void OnEnable()
00160     {
00161         _actions.Enable();
00162         SelectionManager.OnEnemySelectedEvent += EnemySelectedCallBack;
00163         SelectionManager.OnNoSelectionEvent += NoSelectionCallBack;
00164         EnemyHealth.OnEnemyDeadEvent += NoSelectionCallBack;
00165     }
00166
00167     private void OnDisable()
00168     {
00169         _actions.Disable();
00170         SelectionManager.OnEnemySelectedEvent -= EnemySelectedCallBack;
00171         SelectionManager.OnNoSelectionEvent -= NoSelectionCallBack;
00172         EnemyHealth.OnEnemyDeadEvent -= NoSelectionCallBack;
00173     }
00174 }
```

5.63 Player/PlayerExp.cs File Reference

Classes

- class PlayerExp

5.64 PlayerExp.cs

Go to the documentation of this file.

```
00001 using UnityEngine;
00002
00003 public class PlayerExp : MonoBehaviour
00004 {
00005     [Header("Config")]
```

```
00006     [SerializeField] private PlayerStats stats;
00007
00008     private void Update()
00009     {
00010         if (Input.GetKeyDown(KeyCode.X))
00011         {
00012             AddExp(300f);
00013         }
00014     }
00015
00016     public void AddExp(float amount)
00017     {
00018         stats.totalExp += amount;
00019         stats.currentExp += amount;
00020         while (stats.currentExp >= stats.nextLevelExp)
00021         {
00022             stats.currentExp -= stats.nextLevelExp;
00023             NextLevel();
00024         }
00025     }
00026
00027     private void NextLevel()
00028     {
00029         stats.level++;
00030
00031         stats.attributePoints++;
00032
00033         float currentExpRequired = stats.nextLevelExp;
00034         float newNextLevelExp = Mathf.Round(currentExpRequired + stats.nextLevelExp *
      (stats.expMultiplier / 100f));
00035
00036         stats.nextLevelExp = newNextLevelExp;
00037     }
00038 }
```

5.65 Player/PlayerHealth.cs File Reference

Classes

- class PlayerHealth

5.66 PlayerHealth.cs

Go to the documentation of this file.

```
00001 using UnityEngine;
00002
00003 public class PlayerHealth : MonoBehaviour, IDamageable
00004 {
00005     [Header("Config")]
00006     [SerializeField] private PlayerStats stats;
00007
00008     public float CurrentHealth { get; private set; }
00009     private PlayerAnimations _playerAnimations;
00010
00011     private void Awake()
00012     {
00013         _playerAnimations = GetComponent<PlayerAnimations>();
00014     }
00015
00016     private void Update()
00017     {
00018         if (stats.health <= 0f)
00019         {
00020             PlayerDead();
00021         }
00022     }
00023
00024     public void TakeDamage(float amount)
00025     {
00026         if (stats.health <= 0f) return;
00027         stats.health -= amount;
00028         DamageManager.Instance.ShowDamageText(amount, transform);
00029         CurrentHealth = stats.health;
00030         if (!(stats.health <= 0f)) return;
```

```
00031         stats.health = 0f;
00032         PlayerDead();
00033     }
00034
00035     private void PlayerDead()
00036     {
00037         _playerAnimations.SetDeadAnimation();
00038     }
00039 }
```

5.67 Player/PlayerMana.cs File Reference

Classes

- class PlayerMana

5.68 PlayerMana.cs

Go to the documentation of this file.

```
00001 using System;
00002 using UnityEngine;
00003
00004 public class PlayerMana : MonoBehaviour
00005 {
00006     [Header("Config")]
00007     [SerializeField] private PlayerStats stats;
00008
00009     public float CurrentMana { get; private set; }
00010
00011     private void Start()
00012     {
00013         ResetMana();
00014     }
00015
00016     private void Update()
00017     {
00018     }
00019
00020     public void UseMana(float amount)
00021     {
00022         stats.mana = Mathf.Max(stats.mana -= amount, 0f);
00023         CurrentMana = stats.mana;
00024     }
00025
00026     public void ResetMana()
00027     {
00028         CurrentMana = stats.maxMana;
00029     }
00030 }
```

5.69 Player/PlayerMovement.cs File Reference

Classes

- class PlayerMovement

5.70 PlayerMovement.cs

Go to the documentation of this file.

```
00001 using UnityEngine;
00002
00003 public class PlayerMovement : MonoBehaviour
00004 {
00005
00006     [Header("Config")]
00007     [SerializeField] private PlayerStats stats;
00008
00009     public Vector2 MoveDirection => _moveDirection;
00010
00011     private PlayerAnimations _playerAnimations;
00012     private PlayerActions _actions;
00013     private Player _player;
00014     private Rigidbody2D _rb2D;
00015     private Vector2 _moveDirection;
00016
00017     private float _speed;
00018
00019     private void Awake()
00020     {
00021         _player = GetComponent<Player>();
00022         _actions = new PlayerActions();
00023         _rb2D = GetComponent<Rigidbody2D>();
00024         _playerAnimations = GetComponent<PlayerAnimations>();
00025     }
00026
00027     // Update is called once per frame
00028     private void Update()
00029     {
00030         ReadMovement();
00031     }
00032
00033     private void FixedUpdate()
00034     {
00035         Move();
00036     }
00037
00038     private void Move()
00039     {
00040         if (_player.Stats.health <= 0) return;
00041         _rb2D.MovePosition(_rb2D.position + _moveDirection * (stats.speed * Time.fixedDeltaTime));
00042     }
00043     private void ReadMovement()
00044     {
00045         _moveDirection = _actions.Movement.Move.ReadValue<Vector2>().normalized;
00046         if (_moveDirection == Vector2.zero)
00047         {
00048             _playerAnimations.SetMoveBoolTransition(false);
00049             return;
00050         }
00051
00052         _playerAnimations.SetMoveBoolTransition(true);
00053         _playerAnimations.SetMoveAnimation(_moveDirection);
00054     }
00055
00056     private void OnEnable()
00057     {
00058
00059         _actions.Enable();
00060     }
00061
00062     private void OnDisable()
00063     {
00064         _actions.Disable();
00065     }
00066 }
```

5.71 Player/PlayerStats.cs File Reference

Classes

- class PlayerStats

Enumerations

- enum AttributeType { Strength , Agility , Intelligence }

5.71.1 Enumeration Type Documentation

5.71.1.1 AttributeType

```
enum AttributeType
```

Enumerator

Strength	
Agility	
Intelligence	

Definition at line 3 of file PlayerStats.cs.

5.72 PlayerStats.cs

Go to the documentation of this file.

```
00001 using UnityEngine;
00002
00003 public enum AttributeType
00004 {
00005     Strength,
00006     Agility,
00007     Intelligence
00008 }
00009
00010 [CreateAssetMenu(fileName = "PlayerStats", menuName = "Player Stats")]
00011
00012 public class PlayerStats : ScriptableObject
00013 {
00014     [Header("Config")]
00015     public int level;
00016     public float speed;
00017
00018
00019     [Header("Health")]
00020     public float health;
00021     public float maxHealth;
00022
00023     [Header("Mana")]
00024     public float mana;
00025     public float maxMana;
00026
00027     [Header("Exp")]
00028     public float currentExp;
00029     public float nextLevelExp;
00030     public float initialNextLevelExp;
00031     [Range(1f, 100f)] public float expMultiplier;
00032
00033     [Header("Attack")]
00034     public float baseDamage;
00035     public float criticalChance;
00036     public float criticalDamage;
00037
00038     [Header("Attributes")]
00039     public int strength;
00040
00041     public int agility;
00042     public int intelligence;
00043     public int attributePoints;
00044
00045     [HideInInspector] public float totalExp;
00046     [HideInInspector] public float totalDamage;
```

```
00047
00048     public void ResetPlayer()
00049     {
00050         health = maxHealth;
00051         mana = maxMana;
00052         level = 1;
00053         currentExp = 0f;
00054         nextLevelExp = initialNextLevelExp;
00055         totalExp = 0f;
00056         speed = 6f;
00057         baseDamage = 1;
00058         criticalChance = 20;
00059         criticalDamage = 50;
00060         strength = 0;
00061         agility = 0;
00062         intelligence = 0;
00063         attributePoints = 0;
00064     }
00065 }
```

5.73 Player/PlayerUpgrade.cs File Reference

Classes

- class PlayerUpgrade
- class SettingsUpgrade

5.74 PlayerUpgrade.cs

Go to the documentation of this file.

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using UnityEngine;
00005 using UnityEngine.Serialization;
00006
00007 public class PlayerUpgrade : MonoBehaviour
00008 {
00009     public static event Action OnPlayerUpgradeEvent;
00010
00011     [Header("Config")]
00012     [SerializeField] private PlayerStats stats;
00013
00014     [Header("Settings")]
00015     [SerializeField] private SettingsUpgrade[] setting;
00016
00017     private void UpgradePlayer(int upgradeIndex)
00018     {
00019         stats.baseDamage += setting[upgradeIndex].damageUpgrade;
00020         stats.totalDamage += setting[upgradeIndex].damageUpgrade;
00021         stats.maxHealth += setting[upgradeIndex].healthUpgrade;
00022         stats.health = stats.maxHealth;
00023         stats.maxMana += setting[upgradeIndex].manaUpgrade;
00024         stats.mana = stats.maxMana;
00025         stats.criticalChance += setting[upgradeIndex].criticalChanceUpgrade;
00026         stats.criticalDamage += setting[upgradeIndex].criticalDamageUpgrade;
00027         stats.speed += setting[upgradeIndex].speedUpgrade;
00028     }
00029
00030     private void AttributeCallBack(AttributeType attributeType)
00031     {
00032         if (stats.attributePoints == 0) return;
00033         switch (attributeType)
00034         {
00035             case AttributeType.Strength:
00036                 UpgradePlayer(0);
00037                 stats.strength++;
00038                 break;
00039             case AttributeType.Agility:
00040                 UpgradePlayer(1);
00041                 stats.agility++;
00042                 break;
00043             case AttributeType.Intelligence:
00044                 UpgradePlayer(2);
```

```
00045                 stats.intelligence++;
00046                 break;
00047         }
00048
00049         stats.attributePoints--;
00050         OnPlayerUpgradeEvent?.Invoke();
00051     }
00052
00053     private void OnEnable()
00054
00055     {
00056         AttributeButton.OnAttributeSelectedEvent += AttributeCallBack;
00057     }
00058
00059     private void OnDisable()
00060     {
00061         AttributeButton.OnAttributeSelectedEvent -= AttributeCallBack;
00062     }
00063 }
00064
00065 [Serializable]
00066 public class SettingsUpgrade
00067 {
00068     public string name;
00069
00070     [Header("Values")]
00071     public float damageUpgrade;
00072     public float healthUpgrade;
00073     public float manaUpgrade;
00074     public float criticalChanceUpgrade;
00075     public float criticalDamageUpgrade;
00076     public float speedUpgrade;
00077 }
```

5.75 Text/DamageText.cs File Reference

Classes

- class DamageText

5.76 DamageText.cs

Go to the documentation of this file.

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using TMPro;
00004 using UnityEngine;
00005
00006 public class DamageText : MonoBehaviour
00007 {
00008     [Header("Config")]
00009     [SerializeField] private TextMeshProUGUI damageTMP;
00010
00011     public void SetDamageText(float damage)
00012     {
00013         damageTMP.text = damage.ToString();
00014     }
00015
00016     public void DestroyDamageText()
00017     {
00018         Destroy(gameObject);
00019     }
00020 }
```

5.77 Waypoint/Editor/WaypointEditor.cs File Reference

Classes

- class WaypointEditor

5.78 WaypointEditor.cs

Go to the documentation of this file.

```
00001 using System;
00002 using UnityEditor;
00003 using UnityEditor.Graphs;
00004 using UnityEngine;
00005
00006 [CustomEditor(typeof(Waypoint))]
00007 public class WaypointEditor : Editor
00008 {
00009         private Waypoint WaypointTarget => target as Waypoint;
00010
00011         private void OnSceneGUI()
00012         {
00013                 if (WaypointTarget.Points.Length <= 0f) return;
00014
00015                 Handles.color = Color.white;
00016                 for (int i = 0; i < WaypointTarget.Points.Length; i++)
00017                 {
00018                         EditorGUI.BeginChangeCheck();
00019
00020                         Vector3 currentPoint = WaypointTarget.EntityPosition +
      WaypointTarget.Points[i];
00021
00022                         Vector3 newPosition = Handles.FreeMoveHandle(currentPoint, 0.5f, Vector3.one *
      0.5f,
00023                                 Handles.SphereHandleCap);
00024
00025                         GUIStyle text = new GUIStyle();
00026
00027                         text.fontStyle = FontStyle.Bold;
00028                         text.fontSize = 16;
00029                         text.normal.textColor = Color.black;
00030
00031                         Vector3 textPos = new Vector3(0.2f, -0.2f);
00032                         Handles.Label(WaypointTarget.EntityPosition + WaypointTarget.Points[i] +
      textPos, $"{i + 1}", text);
00033
00034                         if (EditorGUI.EndChangeCheck())
00035                         {
00036                                 Undo.RecordObject(target, "Free Move");
00037                                 WaypointTarget.Points[i] = newPosition -
      WaypointTarget.EntityPosition;
00038                         }
00039                 }
00040         }
00041 }
```

5.79 Waypoint/Waypoint.cs File Reference

Classes

- class Waypoint

5.80 Waypoint.cs

Go to the documentation of this file.

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using UnityEngine;
00005
00006 public class Waypoint : MonoBehaviour
00007 {
00008     [Header("Config")]
00009     [SerializeField] private Vector3[] points;
00010
00011     public Vector3[] Points => points;
00012     public Vector3 EntityPosition { get; set; }
00013
00014     private bool _gameStarted;
```

```
00015
00016     private void Start()
00017     {
00018         EntityPosition = transform.position;
00019         _gameStarted = true;
00020     }
00021
00022     public Vector3 GetPosition(int pointIndex)
00023     {
00024         return EntityPosition + points[pointIndex];
00025     }
00026
00027     private void OnDrawGizmos()
00028     {
00029         if (_gameStarted == false && transform.hasChanged)
00030         {
00031             EntityPosition = transform.position;
00032         }
00033     }
00034 }
```

5.81 Weapon/Projectile.cs File Reference

Classes

- class Projectile

5.82 Projectile.cs

Go to the documentation of this file.

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using UnityEngine;
00005
00006 public class Projectile : MonoBehaviour
00007 {
00008     [Header("Config")]
00009     [SerializeField] private float projectileSpeed;
00010
00011     public Vector3 Direction { get; set; }
00012     public float Damage { get; set; }
00013
00014     private void Update()
00015     {
00016         transform.Translate(Direction * (projectileSpeed * Time.deltaTime));
00017     }
00018
00019     private void OnTriggerEnter2D(Collider2D other)
00020     {
00021         other.GetComponent<IDamageable>()?.TakeDamage(Damage);
00022         Destroy(gameObject);
00023     }
00024 }
```

5.83 Weapon/Weapon.cs File Reference

Classes

- class Weapon

Enumerations

- enum WeaponType { Magic , Melee }

5.83.1 Enumeration Type Documentation

5.83.1.1 WeaponType

```
enum WeaponType
```

Enumerator

Magic	
Melee	

Definition at line 6 of file Weapon.cs.

5.84 Weapon.cs

Go to the documentation of this file.

```
00001 using System.Collections;
00002 using System.Collections.Generic;
00003 using UnityEngine;
00004 using UnityEngine.Serialization;
00005
00006 public enum WeaponType
00007 {
00008     Magic,
00009     Melee
00010 }
00011
00012 [CreateAssetMenu(fileName = "Weapon_")]
00013 public class Weapon : ScriptableObject
00014 {
00015     [Header("Config")]
00016     public Sprite icon;
00017     public WeaponType weaponType;
00018     public float damage;
00019
00020     [Header("Projectile")]
00021     public Projectile projectilePrefab;
00022     public float requiredMana;
00023
00024 }
```

5.85 Weapon/WeaponChange.cs File Reference

Classes

- class WeaponChange

5.86 WeaponChange.cs

Go to the documentation of this file.

```
00001 using System;
00002 using System.Collections;
00003 using System.Collections.Generic;
00004 using TMPro;
00005 using UnityEngine;
00006 using UnityEngine.UI;
00007
00008 public class WeaponChange : MonoBehaviour
00009 {
```

```
00010     [Header("Config")]
00011     [SerializeField] private Player player;
00012
00013     [SerializeField] private Image weaponIcon;
00014
00015     private Weapon[] _weapons;
00016     private int _weaponIndex;
00017
00018     private void Awake()
00019     {
00020         _weaponIndex = 0;
00021         _weapons = player.GetComponent<PlayerAttack>().allWeapons;
00022     }
00023
00024     public void ChangeWeapon()
00025     {
00026         _weaponIndex++;
00027         if (_weaponIndex >= _weapons.Length)
00028         {
00029             _weaponIndex = 0;
00030         }
00031
00032         var weapon = _weapons[_weaponIndex];
00033         weaponIcon.sprite = weapon.icon;
00034
00035         player.GetComponent<PlayerAttack>().EquipWeapon(weapon);
00036     }
00037 }
00038
```

Index