

Namespace Name

Classes

[AvoidCarScript](#)

Class AvoidCarScript

Namespace: [Name](#)

Assembly: SamuraiSoccer.StageContents.UK.dll

```
public class AvoidCarScript : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← AvoidCarScript

Methods

Start()

```
private void Start()
```

Namespace SamuraiSoccer

Classes

[DetectExceptionLog](#)

[InFileTransmitClient<T>](#)

セーブデータ等、保存するデータの受け渡しを行う。

[InMemoryDataTransitClient<T>](#)

シーン間のデータ受け渡し等のデータ伝送に使用。保存はされない。

[InNetworkTransmitClient<T>](#)

サーバーとデータを送受信する。

[MeshCut](#)

[MeshCut.MeshData](#)

[MeshCut.RectangleFragment](#)

[MeshCut.TriangleFragment](#)

[ModifyStageData](#)

[SaveData](#)

[SoundMaster](#)

[StorageKey](#)

保存・一時保存のデータ受け渡しに使うキー

Interfaces

[IDataTransmitClient<T>](#)

データを受け渡し・送受信するためのインターフェース。

Class DetectExceptionLog

Namespace: [SamuraiSoccer](#)

Assembly: SamuraiSoccer.dll

```
public class DetectExceptionLog : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← DetectExceptionLog

Fields

m_logText

```
[SerializeField]  
private Text m_logText
```

Field Value

Text

Methods

HandleLog(string, string, LogType)

```
private void HandleLog(string logString, string stackTrace, LogType type)
```

Parameters

logString [string](#)

stackTrace [string](#)

type LogType

OnDisable()

```
private void OnDisable()
```

OnEnable()

```
private void OnEnable()
```

Interface **IDataTransmitClient<T>**

Namespace: [SamuraiSoccer](#)

Assembly: SamuraiSoccer.dll

データを受け渡し・送受信するためのインターフェース。

```
public interface IDataTransmitClient<T>
```

Type Parameters

T

やり取りするデータの型名。

Methods

Get(string)

データを受け取り・受信する。

```
T Get(string key)
```

Parameters

key [string](#)

渡すときと共通のキーまたはURL。

Returns

T

受け取ったデータ。

GetAsync(string)

データを受け取り・受信する。

```
UniTask<T> GetAsync(string key)
```

Parameters

key string ↗

渡すときと共通のキーまたはURL.

Returns

```
UniTask<T>
```

受け取ったデータ.

Set(string, T)

データを渡す・送信する.

```
void Set(string key, T value)
```

Parameters

key string ↗

受け取るときと共通のキーまたはURL.

value T

送るデータ.

SetAsync(string, T)

データを渡す・送信する.

```
UniTask SetAsync(string key, T value)
```

Parameters

key [string](#)

受け取るときと共通のキーまたはURL。

value T

送るデータ。

Returns

UniTask

TryGet(string, out T)

データを受け取り・受信する。

```
bool TryGet(string key, out T value)
```

Parameters

key [string](#)

渡すときと共通のキーまたはURL。

value T

受け取ったデータ。

Returns

[bool](#)

データの受け取り・受信に成功したか。

Class InFileTransmitClient<T>

Namespace: [SamuraiSoccer](#)

Assembly: SamuraiSoccer.dll

セーブデータ等、保存するデータの受け渡しを行う。

```
public class InFileTransmitClient<T> : IDataTransmitClient<T>
```

Type Parameters

T

受け渡しするデータの型。

Inheritance

[object](#) ↗ ← InFileTransmitClient<T>

Implements

[IDataTransmitClient](#)<T>

Fields

FolderPath

```
private readonly stringFolderPath
```

Field Value

[string](#) ↗

typeName

```
private readonly string typeName
```

Field Value

[string](#)

Methods

Get(string)

データを受け取り・受信する。

```
public T Get(string key)
```

Parameters

key [string](#)

渡すときと共通のキーまたはURL。

Returns

T

受け取ったデータ。

GetAsync(string)

データを受け取り・受信する。

```
public UniTask<T> GetAsync(string key)
```

Parameters

key [string](#)

渡すときと共通のキーまたはURL。

Returns

UniTask<T>

受け取ったデータ。

GetSavePath(string)

```
private string GetSavePath(string key)
```

Parameters

key [string](#)

Returns

[string](#)

Set(string, T)

データを渡す・送信する。

```
public void Set(string key, T value)
```

Parameters

key [string](#)

受け取るときと共通のキーまたはURL。

value T

送るデータ。

SetAsync(string, T)

データを渡す・送信する。

```
public UniTask SetAsync(string key, T value)
```

Parameters

key [string](#)

受け取るときと共通のキーまたはURL。

value T

送るデータ。

Returns

UniTask

ThrowIfInvalidType()

```
private void ThrowIfInvalidType()
```

TryGet(string, out T)

データを受け取り・受信する。

```
public bool TryGet(string key, out T value)
```

Parameters

key [string](#)

渡すときと共通のキーまたはURL。

value T

受け取ったデータ。

Returns

[bool](#)

データの受け取り・受信に成功したか。

Class InMemoryDataTransitClient<T>

Namespace: [SamuraiSoccer](#)

Assembly: SamuraiSoccer.dll

シーン間のデータ受け渡し等のデータ伝送に使用。保存はされない。

```
public class InMemoryDataTransitClient<T> : IDataTransmitClient<T>
```

Type Parameters

T

受け渡しするデータ型。

Inheritance

[object](#) ← InMemoryDataTransitClient<T>

Implements

[IDataTransmitClient](#)<T>

Fields

strage

```
private static readonly Dictionary<string, T> strage
```

Field Value

[Dictionary](#)<[string](#), T>

Methods

Get(string)

データを受け取り・受信する。

```
public T Get(string key)
```

Parameters

key [string](#) ↗

渡すときと共通のキーまたはURL.

Returns

T

受け取ったデータ.

GetAsync(string)

データを受け取り・受信する.

```
public UniTask<T> GetAsync(string key)
```

Parameters

key [string](#) ↗

渡すときと共通のキーまたはURL.

Returns

UniTask<T>

受け取ったデータ.

Set(string, T)

データを渡す・送信する.

```
public void Set(string key, T value)
```

Parameters

key [string](#) ↗

受け取るときと共通のキーまたはURL。

value T

送るデータ。

SetAsync(string, T)

データを渡す・送信する。

```
public UniTask SetAsync(string key, T value)
```

Parameters

key [string](#) ↗

受け取るときと共通のキーまたはURL。

value T

送るデータ。

Returns

UniTask

TryGet(string, out T)

データを受け取り・受信する。

```
public bool TryGet(string key, out T value)
```

Parameters

key [string](#) ↗

渡すときと共通のキーまたはURL。

value T

受け取ったデータ。

Returns

bool ↗

データの受け取り・受信に成功したか。

Class InNetworkTransmitClient<T>

Namespace: [SamuraiSoccer](#)

Assembly: SamuraiSoccer.dll

サーバーとデータを送受信する。

```
public class InNetworkTransmitClient<T> : IDataTransmitClient<T>
```

Type Parameters

T

受け渡しするデータの型。

Inheritance

[object](#) ← InNetworkTransmitClient<T>

Implements

[IDataTransmitClient](#)<T>

Constructors

InNetworkTransmitClient()

```
public InNetworkTransmitClient()
```

InNetworkTransmitClient(HttpRequestHeaders)

デフォルトのヘッダを指定する際のコンストラクタ。

```
public InNetworkTransmitClient(HttpRequestHeaders defaultHeader)
```

Parameters

defaultHeader [HttpRequestHeaders](#)

「デフォルトのヘッダ」

Fields

client

```
private readonly HttpClient client
```

Field Value

[HttpClient](#)

Methods

Get(string)

使用できません。

```
public T Get(string key)
```

Parameters

key [string](#)

Returns

T

GetAsync(string)

データを受け取り・受信する。

```
public UniTask<T> GetAsync(string key)
```

Parameters

key [string](#)

渡すときと共通のキーまたはURL。

Returns

[UniTask<T>](#)

受け取ったデータ。

Set(string, T)

使用できません。

```
public void Set(string key, T value)
```

Parameters

key [string](#)

value [T](#)

SetAsync(string, T)

データを渡す・送信する。

```
public UniTask SetAsync(string key, T value)
```

Parameters

key [string](#)

受け取るときと共通のキーまたはURL。

value [T](#)

送るデータ。

Returns

TryGet(string, out T)

使用できません。

```
public bool TryGet(string key, out T value)
```

Parameters

key [string](#)

value [T](#)

Returns

[bool](#)

Class MeshCut

Namespace: [SamuraiSoccer](#)

Assembly: SamuraiSoccer.dll

```
public class MeshCut : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← MeshCut

Fields

_backMeshData

```
private static MeshCut.MeshData _backMeshData
```

Field Value

[MeshCut.MeshData](#)

_frontMeshData

```
private static MeshCut.MeshData _frontMeshData
```

Field Value

[MeshCut.MeshData](#)

_isFront

```
private static bool[] _isFront
```

Field Value

`bool` []

_slashPlane

```
private static Plane _slashPlane
```

Field Value

Plane

_targetMesh

```
private static Mesh _targetMesh
```

Field Value

Mesh

_targetNormals

```
private static Vector3[] _targetNormals
```

Field Value

Vector3[]

_targetUVs

```
private static Vector2[] _targetUVs
```

Field Value

Vector2[]

`_targetVertices`

```
private static Vector3[] _targetVertices
```

Field Value

Vector3[]

Methods

CutMesh(GameObject, Vector3, Vector3)

gameObjectを切断して2つのMeshにして返します。1つ目のMeshが切断面の法線に対して表側、2つ目が裏側です。
何度も切るようなオブジェクトでも頂点数が増えないように処理をしてあります

```
public static Mesh[] CutMesh(GameObject target, Vector3 planeAnchorPoint,  
Vector3 planeNormalDirection)
```

Parameters

target GameObject

切断対象のgameObject

planeAnchorPoint Vector3

切断面上の1点

planeNormalDirection Vector3

切断面の法線

Returns

Mesh[]

CutMeshOnce(GameObject, Vector3, Vector3)

gameObjectを切断して2つのMeshにして返します。1つ目のMeshが切断面の法線に対して表側、2つ目が裏側です。
切断の処理が雑なので何度も斬ると頂点数が膨れ上がります。そのかわり一回斬るだけならこっちが早い

```
public static Mesh[] CutMeshOnce(GameObject target, Vector3 planeAnchorPoint,  
Vector3 planeNormalDirection)
```

Parameters

target GameObject

切断対象のgameObject

planeAnchorPoint Vector3

切断面上の1点

planeNormalDirection Vector3

切断面の法線

Returns

Mesh[]

CutMeshRough(GameObject, Vector3, Vector3)

gameObjectを雑に切断して2つのMeshにして返します。切断面がザラザラなのでハイポリにしか使えませんが処理はちょっと早いです。1つ目のMeshが切断面の法線に対して表側、2つ目が裏側です。

```
public static Mesh[] CutMeshRough(GameObject target, Vector3 planeAnchorPoint,  
Vector3 planeNormalDirection)
```

Parameters

target GameObject

切断対象のgameObject

planeAnchorPoint Vector3

切断面上の1点

`planeNormalDirection` Vector3

切断面の法線

Returns

Mesh[]

Sepalate(bool[], int[], int)

```
private static void Sepalate(bool[] sides, int[] vertexIndices, int submesh)
```

Parameters

`sides` [bool](#)[]

`vertexIndices` [int](#)[]

`submesh` [int](#)

SepalateOnce(bool[], int[], int)

```
private static void SepalateOnce(bool[] sides, int[] vertexIndices, int submesh)
```

Parameters

`sides` [bool](#)[]

`vertexIndices` [int](#)[]

`submesh` [int](#)

Class MeshCut.MeshData

Namespace: [SamuraiSoccer](#)

Assembly: SamuraiSoccer.dll

```
public class MeshCut.MeshData
```

Inheritance

[object](#) ← MeshCut.MeshData

Fields

normals

```
public List<Vector3> normals
```

Field Value

[List](#)<Vector3>

rectangleFragments

```
public List<List<MeshCut.RectangleFragment>> rectangleFragments
```

Field Value

[List](#)<[List](#)<[MeshCut.RectangleFragment](#)>>

subMeshIndices

```
public List<List<int>> subMeshIndices
```

Field Value

[List](#) <[List](#) <[int](#)>>

threshold

`private const float threshold = 0.0001`

Field Value

[float](#)

trackNum

`private int trackNum`

Field Value

[int](#)

trackedArray

`private int[] trackedArray`

Field Value

[int](#)[]

trackedVertexNum

`private int trackedVertexNum`

Field Value

[int](#)

triangleFragments

```
public List<List<MeshCut.TriangleFragment>> triangleFragments
```

Field Value

[List](#) <[List](#) <[MeshCut.TriangleFragment](#)>>

triangles

```
public List<int> triangles
```

Field Value

[List](#) <[int](#)>

UVS

```
public List<Vector2> uvs
```

Field Value

[List](#) <[Vector2](#)>

vertices

```
public List<Vector3> vertices
```

Field Value

[List](#) <[Vector3](#)>

verticesArray

```
public Vector3[] verticesArray
```

Field Value

Vector3[]

Methods

AddTriangle(int, int, int, int)

```
public void AddTriangle(int p1, int p2, int p3, int submeshNum)
```

Parameters

p1 [int](#)

p2 [int](#)

p3 [int](#)

submeshNum [int](#)

AddTriangle(int, int, Vector3, Vector3, Vector2, int)

```
public void AddTriangle(int notCutIndex0, int notCutIndex1, Vector3 cutPoint, Vector3
normal, Vector2 uv, int submeshNum)
```

Parameters

notCutIndex0 [int](#)

notCutIndex1 [int](#)

cutPoint Vector3

normal Vector3

uv Vector2

`submeshNum` [int ↗](#)

AddTriangle(int, Vector3, Vector3, Vector3[], Vector2[], int)

```
public void AddTriangle(int notCutIndex, Vector3 cutPoint1, Vector3 cutPoint2, Vector3[]  
normals3, Vector2[] uvs3, int submeshNum)
```

Parameters

`notCutIndex` [int ↗](#)

`cutPoint1` Vector3

`cutPoint2` Vector3

`normals3` Vector3[]

`uvs3` Vector2[]

`submeshNum` [int ↗](#)

CheckIndex(int)

```
private bool CheckIndex(int target)
```

Parameters

`target` [int ↗](#)

Returns

[bool ↗](#)

ClearAll()

```
public void ClearAll()
```

ConnectFragments()

```
public void ConnectFragments()
```

Class MeshCut.RectangleFragment

Namespace: [SamuraiSoccer](#)

Assembly: SamuraiSoccer.dll

```
public class MeshCut.RectangleFragment
```

Inheritance

[object](#) ← MeshCut.RectangleFragment

Constructors

RectangleFragment(int, int, Vector3, Vector3, Vector2[],
Vector3[])

```
public RectangleFragment(int notCutPoint1Index, int notCutPoint2Index, Vector3 cutPoint1,  
Vector3 cutPoint2, Vector2[] uvs, Vector3[] normals)
```

Parameters

notCutPoint1Index [int](#)

notCutPoint2Index [int](#)

cutPoint1 Vector3

cutPoint2 Vector3

uvs Vector2[]

normals Vector3[]

Fields

cutLine

```
public Vector3 cutLine
```

Field Value

Vector3

cutPoints

```
public Vector3[] cutPoints
```

Field Value

Vector3[]

normals

```
public Vector3[] normals
```

Field Value

Vector3[]

notCutPointIndecies

```
public int[] notCutPointIndecies
```

Field Value

[int](#)[]

UVS

```
public Vector2[] uvs
```

Field Value

Vector2[]

Class MeshCut.TriangleFragment

Namespace: [SamuraiSoccer](#)

Assembly: SamuraiSoccer.dll

```
public class MeshCut.TriangleFragment
```

Inheritance

[object](#) ← MeshCut.TriangleFragment

Constructors

TriangleFragment(int, Vector3, Vector3, Vector2[], Vector3[])

```
public TriangleFragment(int notCutPoint1Index, Vector3 cutPoint1, Vector3 cutPoint2,  
Vector2[] uvs, Vector3[] normals)
```

Parameters

notCutPoint1Index [int](#)

cutPoint1 Vector3

cutPoint2 Vector3

uvs Vector2[]

normals Vector3[]

Fields

cutLine

```
public Vector3 cutLine
```

Field Value

Vector3

cutPoints

```
public Vector3[] cutPoints
```

Field Value

Vector3[]

normals

```
public Vector3[] normals
```

Field Value

Vector3[]

notCutPointIndecies

```
public int notCutPointIndecies
```

Field Value

[int↗](#)

UVS

```
public Vector2[] uvs
```

Field Value

Vector2[]

Class ModifyStageData

Namespace: [SamuraiSoccer](#)

Assembly: ScriptsEditor.dll

```
public class ModifyStageData : EditorWindow
```

Inheritance

[Object](#) ← Object ← ScriptableObject ← EditorWindow ← ModifyStageData

Fields

m_data

```
private SaveData m_data
```

Field Value

[SaveData](#)

Methods

OnGUI()

```
private void OnGUI()
```

ShowWindow()

```
[MenuItem("Custom/Modify stage data")]
private static void ShowWindow()
```

Class SaveData

Namespace: [SamuraiSoccer](#)

Assembly: SamuraiSoccer.dll

```
[Serializable]
public class SaveData
```

Inheritance

[object](#) ← SaveData

Fields

m_stageData

```
public int m_stageData
```

Field Value

[int](#)

Class SoundMaster

Namespace: [SamuraiSoccer](#)

Assembly: SamuraiSoccer.dll

```
public class SoundMaster : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← SoundMaster

Fields

bgm AudioSource

```
private static AudioSource bgm
```

Field Value

AudioSource

bgmBolumne

```
private float bgmBolumne
```

Field Value

[float](#)

bgmSelectedIndex

```
private int bgmSelectedIndex
```

Field Value

[int](#)

instance

```
private static SoundMaster instance
```

Field Value

[SoundMaster](#)

se AudioSource

```
private static AudioSource se AudioSource
```

Field Value

AudioSource

se Bolum

```
public float se Bolum
```

Field Value

[float](#)

sound Database

```
private static SoundDatabase sound Database
```

Field Value

[SoundDatabase](#)

Properties

BGMVolume

```
public float BGMVolume { get; set; }
```

Property Value

[float](#)

Instance

```
public static SoundMaster Instance { get; }
```

Property Value

[SoundMaster](#)

Methods

OnDestroy()

```
private void OnDestroy()
```

PlayBGM(int)

BGMを流す

```
public void PlayBGM(int soundIndex)
```

Parameters

soundIndex [int](#)

音源番号

PlaySE(int)

SEを流す

```
public UniTask PlaySE(int soundIndex)
```

Parameters

soundIndex [int](#)

音源番号

Returns

UniTask

Class StorageKey

Namespace: [SamuraiSoccer](#)

Assembly: SamuraiSoccer.dll

保存・一時保存のデータ受け渡しに使うキー

```
public class StorageKey
```

Inheritance

[object](#) ↗ ← StorageKey

Fields

KEY_FIELDNUMBER

```
public const string KEY_FIELDNUMBER = "KEY_FIELDNUMBER"
```

Field Value

[string](#) ↗

KEY_GROUNDNUMBER

```
public const string KEY_GROUNDNUMBER = "KEY_GROUNDNUMBER"
```

Field Value

[string](#) ↗

KEY OPPONENT_TYPE

```
public const string KEY OPPONENT_TYPE = "KEY OPPONENT_TYPE"
```

Field Value

[string](#) ↗

KEY_RESULTMESSAGE

```
public const string KEY_RESULTMESSAGE = "KEY_RESULTMESSAGE"
```

Field Value

[string](#) ↗

KEY_STAGENUMBER

```
public const string KEY_STAGENUMBER = "KEY_STAGENUMBER"
```

Field Value

[string](#) ↗

KEY_WINORLOSE

```
public const string KEY_WINORLOSE = "KEY_WINORLOSE"
```

Field Value

[string](#) ↗

Namespace SamuraiSoccer.AI.EditorExtension

Classes

[FieldMaker](#)

フィールドを作るためのクラス。メニューインストラウに出るけど実行用。スクリプトで内容を書く。

Class FieldMaker

Namespace: [SamuraiSoccer.AI.EditorExtension](#)

Assembly: SamuraiSoccer.SoccerGame.AI.EditorExtension.dll

フィールドを作るためのクラス。メニューインドウに出るけど実行用。スクリプトで内容を書く。

```
public class FieldMaker : EditorWindow
```

Inheritance

[Object](#) ← Object ← ScriptableObject ← EditorWindow ← FieldMaker

Methods

ClickButton()

```
[MenuItem("Custom/FieldMaker")]
private static void ClickButton()
```

Namespace SamuraiSoccer.Event

Classes

[InGameEvent](#)

試合中のイベントを管理

[PlayerEvent](#)

プレイヤーのイベントを管理

[StageSelectEvent](#)

[UIEffectEvent](#)

Enums

[GoalEventType](#)

ゴールイベントの種類。

[Stage](#)

ステージ(国名or全体)

Enum GoalEventType

Namespace: [SamuraiSoccer.Event](#)

Assembly: SamuraiSoccer.Event.dll

ゴールイベントの種類。

```
public enum GoalEventType
```

Fields

CutSceneOpponentGoal = 3

CutSceneTeammateGoal = 2

NormalOpponentGoal = 1

NormalTeammateGoal = 0

Class InGameEvent

Namespace: [SamuraiSoccer.Event](#)

Assembly: SamuraiSoccer.Event.dll

試合中のイベントを管理

```
public static class InGameEvent
```

Inheritance

[object](#) ← InGameEvent

Fields

m_finishShareObservable

```
private static IObservable<Unit> m_finishShareObservable
```

Field Value

[IObservable](#)<Unit>

m_finishSubject

```
private static Subject<Unit> m_finishSubject
```

Field Value

Subject<Unit>

m_goalShareObservable

```
private static IObservable<GoalEventType> m_goalShareObservable
```

Field Value

[IObservable](#) <[GoalEventType](#)>

m_goalSubject

```
private static Subject<GoalEventType> m_goalSubject
```

Field Value

Subject<[GoalEventType](#)>

m_isPlaying

```
private static bool m_isPlaying
```

Field Value

[bool](#)

m_opponentScoreSubject

```
private static ReactiveProperty<int> m_opponentScoreSubject
```

Field Value

ReactiveProperty<[int](#)>

m_pauseShareObservable

```
private static IObservable<bool> m_pauseShareObservable
```

Field Value

[IObservable](#)<[bool](#)>

m_pauseSubject

```
private static Subject<bool> m_pauseSubject
```

Field Value

Subject<[bool](#)>

m_penaltySubject

```
private static Subject<int> m_penaltySubject
```

Field Value

Subject<[int](#)>

m_penaltySubjectObservable

```
private static IObservable<int> m_penaltySubjectObservable
```

Field Value

[IObservable](#)<[int](#)>

m_playShareObservable

```
private static IObservable<Unit> m_playShareObservable
```

Field Value

[IObservable](#)<Unit>

m_playSubject

```
private static Subject<Unit> m_playSubject
```

Field Value

Subject<Unit>

m_resetSubject

```
private static ReplaySubject<Unit> m_resetSubject
```

Field Value

ReplaySubject<Unit>

m_standbyShareObservable

```
private static IObservable<bool> m_standbyShareObservable
```

Field Value

[IObservable](#)<[bool](#)>

m_standbySubject

```
private static Subject<bool> m_standbySubject
```

Field Value

Subject<[bool](#)>

m_teammateScoreSubject

```
private static ReactiveProperty<int> m_teammateScoreSubject
```

Field Value

ReactiveProperty<[int](#)>

m_updateDuringPlayObservable

```
private static IObservable<long> m_updateDuringPlayObservable
```

Field Value

[IObservable](#)<[long](#)>

Properties

Finish

FinishイベントのSubscribe先

```
public static IObservable<Unit> Finish { get; }
```

Property Value

[IObservable](#)<Unit>

Goal

GoalイベントのSubscribe先

```
public static IObservable<GoalEventType> Goal { get; }
```

Property Value

[IObservable](#)<[GoalEventType](#)>

OpponentScore

OpponentScoreイベントのSubscribe先

```
public static IObservable<int> OpponentScore { get; }
```

Property Value

[IObservable](#)<[int](#)>

Pause

PauseイベントのSubscribe先 true:一時停止, false:解除

```
public static IObservable<bool> Pause { get; }
```

Property Value

[IObservable](#)<[bool](#)>

Penalty

PenaltyイベントのSubscribe先 0:警告, 1:退場

```
public static IObservable<int> Penalty { get; }
```

Property Value

[IObservable](#)<[int](#)>

Play

PlayイベントのSubscribe先

```
public static IObservable<Unit> Play { get; }
```

Property Value

[IObservable](#)<Unit>

Reset

ResetイベントのSubscribe先

```
public static IObservable<Unit> Reset { get; }
```

Property Value

[IObservable](#)<Unit>

Standby

StandbyイベントのSubscribe先

```
public static IObservable<bool> Standby { get; }
```

Property Value

[IObservable](#)<bool>

TeammateScore

TeammateScoreイベントのSubscribe先

```
public static IObservable<int> TeammateScore { get; }
```

Property Value

[IObservable](#)<int>

UpdateDuringPlay

Play中にUpdateタイミングで発行され続けるストリーム

```
public static IObservable<long> UpdateDuringPlay { get; }
```

Property Value

[IObservable](#)<[long](#)>

Methods

FinishOnNext()

Finishイベントの発行

```
public static void FinishOnNext()
```

GoalOnNext(GoalEventType)

Goalイベントの発行

```
public static void GoalOnNext(GoalEventType type)
```

Parameters

type [GoalEventType](#)

OpponentScoreOnNext(int)

OpponentScoreイベントの発行

```
public static void OpponentScoreOnNext(int score)
```

Parameters

score int

PauseOnNext(bool)

Pauseイベントの発行

```
public static void PauseOnNext(bool isPause)
```

Parameters

isPause bool

true:一時停止, false:解除

PenaltyOnNext(int)

Penaltyイベントの発行

```
public static void PenaltyOnNext(int penaltycount)
```

Parameters

penaltycount int

0:警告, 1:退場

PlayOnNext()

Playイベントを発行

```
public static void PlayOnNext()
```

ResetOnNext()

Resetのイベントを発行

```
public static void ResetOnNext()
```

ResetResetSubject()

```
public static void ResetResetSubject()
```

StandbyOnNext(bool)

Standbyイベントを発行

```
public static void StandbyOnNext(bool isTeammateBall = false)
```

Parameters

isTeammateBall [bool](#)

自チームボールか。

TeammateScoreOnNext(int)

TeammateScoreイベントの発行

```
public static void TeammateScoreOnNext(int score)
```

Parameters

score [int](#)

Class PlayerEvent

Namespace: [SamuraiSoccer.Event](#)

Assembly: SamuraiSoccer.Event.dll

プレイヤーのイベントを管理

```
public static class PlayerEvent
```

Inheritance

[object](#) ← PlayerEvent

Fields

m_attackShareObservable

```
private static IObservable<Unit> m_attackShareObservable
```

Field Value

[IObservable](#)<Unit>

m_attackSubject

```
private static Subject<Unit> m_attackSubject
```

Field Value

Subject<Unit>

m_faulCheckShareObservable

```
private static IObservable<Unit> m_faulCheckShareObservable
```

Field Value

[IObservable](#) <Unit>

m_faulCheckSubject

```
private static Subject<Unit> m_faulCheckSubject
```

Field Value

Subject<Unit>

m_getPenalty

```
private static IReadOnlyReactiveProperty<bool> m_getPenalty
```

Field Value

IReadOnlyReactiveProperty<[bool](#)>

m_isEnableAttack

```
private static IReadOnlyReactiveProperty<bool> m_isEnableAttack
```

Field Value

IReadOnlyReactiveProperty<[bool](#)>

m_isEnableChargeAttack

```
private static ReactiveProperty<bool> m_isEnableChargeAttack
```

Field Value

ReactiveProperty<[bool](#)>

m_isInChargeAttack

```
private static ReactiveProperty<bool> m_isInChargeAttack
```

Field Value

ReactiveProperty<[bool](#)>

m_lockChargeAttack

```
private static ReactiveProperty<bool> m_lockChargeAttack
```

Field Value

ReactiveProperty<[bool](#)>

m_stickDir

```
private static ReactiveProperty<Vector3> m_stickDir
```

Field Value

ReactiveProperty<Vector3>

m_stickInputShareObservable

```
private static IObservable<Vector3> m_stickInputShareObservable
```

Field Value

[IObservable](#)<Vector3>

m_stickInputSubject

```
private static Subject<Vector3> m_stickInputSubject
```

Field Value

Subject<Vector3>

Properties

Attack

AttackイベントのSubscribe先

```
public static IObservable<Unit> Attack { get; }
```

Property Value

[IObservable](#)<Unit>

FaulCheck

```
public static IObservable<Unit> FaulCheck { get; }
```

Property Value

[IObservable](#)<Unit>

IsEnableChargeAttack

ため攻撃が可能かどうか

```
public static IReadOnlyReactiveProperty<bool> IsEnableChargeAttack { get; }
```

Property Value

IReadOnlyReactiveProperty<bool>

IsInChargeAttack

ため攻撃中かどうか

```
public static IReadOnlyReactiveProperty<bool> IsInChargeAttack { get; }
```

Property Value

IReadOnlyReactiveProperty<bool>

IsLockChargeAttack

ため攻撃を使えなくするかどうか

```
public static IReadOnlyReactiveProperty<bool> IsLockChargeAttack { get; }
```

Property Value

IReadOnlyReactiveProperty<bool>

StickDir

```
public static IReadOnlyReactiveProperty<Vector3> StickDir { get; }
```

Property Value

IReadOnlyReactiveProperty<Vector3>

StickInput

移動イベントを発行

```
public static IObservable<Vector3> StickInput { get; }
```

Property Value

[IObservable](#)<Vector3>

Methods

AttackOnNext()

Attackのイベントを発行

```
public static void AttackOnNext()
```

FaulCheckOnNext()

```
public static void FaulCheckOnNext()
```

SetIsEnableChargeAtack(bool)

```
public static void SetIsEnableChargeAtack(bool flag)
```

Parameters

[flag](#) [bool](#)

SetIsInChargeAtack(bool)

```
public static void SetIsInChargeAtack(bool flag)
```

Parameters

[flag](#) [bool](#)

SetLockChargeAttack(bool)

ため攻撃の使用可能状態を設定する

```
public static void SetLockChargeAttack(bool value)
```

Parameters

value bool

true : ため攻撃できない

StickControllerOnNext(Vector3)

```
public static void StickControllerOnNext(Vector3 dir)
```

Parameters

dir Vector3

Enum Stage

Namespace: [SamuraiSoccer.Event](#)

Assembly: SamuraiSoccer.Event.dll

ステージ(国名or全体)

```
public enum Stage
```

Fields

China = 2

Japan = 5

Russian = 4

UK = 1

USA = 3

World = 0

Class StageSelectEvent

Namespace: [SamuraiSoccer.Event](#)

Assembly: SamuraiSoccer.Event.dll

```
public static class StageSelectEvent
```

Inheritance

[object](#) ← StageSelectEvent

Fields

m_preview

```
private static Subject<int> m_preview
```

Field Value

[Subject<int>](#)

m_previewShareObservable

```
private static IObservable<int> m_previewShareObservable
```

Field Value

[IObservable<int>](#)

m_stage

```
private static Subject<Stage> m_stage
```

Field Value

Subject<[Stage](#)>

m_stageShareObservable

```
private static IObservable<Stage> m_stageShareObservable
```

Field Value

[IObservable](#)<[Stage](#)>

Properties

Preview

Previewイベントを発行

```
public static IObservable<int> Preview { get; }
```

Property Value

[IObservable](#)<[int](#)>

Stage

StageイベントのSubscribe先

```
public static IObservable<Stage> Stage { get; }
```

Property Value

[IObservable](#)<[Stage](#)>

Methods

PreviewOnNext(int)

PreviewイベントのSubscribe先

```
public static void PreviewOnNext(int number)
```

Parameters

number [int](#)

ステージ番号

StageOnNext(Stage)

Stageイベントを発行

```
public static void StageOnNext(Stage stage)
```

Parameters

stage [Stage](#)

ステージ(国名)

Class UIEffectEvent

Namespace: [SamuraiSoccer.Event](#)

Assembly: SamuraiSoccer.Event.dll

```
public static class UIEffectEvent
```

Inheritance

[object](#) ← UIEffectEvent

Fields

m_blackOutShareObservable

```
private static IObservable<float> m_blackOutShareObservable
```

Field Value

[IObservable](#)<[float](#)>

m_blackOutSubject

```
private static Subject<float> m_blackOutSubject
```

Field Value

[Subject](#)<[float](#)>

Properties

BlackOut

UI暗転イベントのSubscribe先

```
public static IObservable<float> BlackOut { get; }
```

Property Value

[IObservable](#)<[float](#)>

Methods

BlackOutOnNext(float)

UI暗転イベントを発行

```
public static void BlackOutOnNext(float totalsec)
```

Parameters

totalsec [float](#)

暗転の全体時間

Namespace SamuraiSoccer.Event.Editor Extension

Classes

[InGameEventEditor](#)

Class InGameEventEditor

Namespace: [SamuraiSoccer.Event.EditorExtension](#)

Assembly: SamuraiSoccer.Event.EditorExtension.dll

```
public class InGameEventEditor : EditorWindow
```

Inheritance

[object](#) ← Object ← ScriptableObject ← EditorWindow ← InGameEventEditor

Fields

m_hasYellowCard

```
private bool m_hasYellowCard
```

Field Value

[bool](#)

m_isPaused

```
private bool m_isPaused
```

Field Value

[bool](#)

Methods

OnGUI()

```
private void OnGUI()
```

ShowWindow()

```
[MenuItem("Custom/InGameEventDebugger")]
private static void ShowWindow()
```

Namespace SamuraiSoccer.Player

Classes

[AnimationController](#)

[GoalCut](#)

ゴールを斬撃で切れるように

[PlayerAttack](#)

[PlayerEffect](#)

[PlayerMove](#)

[Slash](#)

[TESTCHARGEATTACK](#)

[Trail](#)

Enums

[PlayerMove.State](#)

Class AnimationController

Namespace: [SamuraiSoccer.Player](#)

Assembly: SamuraiSoccer.Player.dll

```
public class AnimationController : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← AnimationController

Fields

animator

```
private Animator animator
```

Field Value

Animator

m_isEnableAttack

```
private bool m_isEnableAttack
```

Field Value

[bool](#)

m_isIdle

```
private bool m_isIdle
```

Field Value

bool ↗

m_isRunning

```
private bool m_isRunning
```

Field Value

bool ↗

Methods

Attack()

```
public void Attack()
```

Run()

```
public void Run()
```

Start()

```
private void Start()
```

Stay()

```
public void Stay()
```

Class GoalCut

Namespace: [SamuraiSoccer.Player](#)

Assembly: SamuraiSoccer.Player.dll

ゴールを斬撃で切れるように

```
public class GoalCut : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← GoalCut

Fields

m_cutted

```
private bool m_cutted
```

Field Value

[bool](#)

Methods

Cut()

```
private void Cut()
```

OnTriggerEnter(Collider)

```
private void OnTriggerEnter(Collider other)
```

Parameters

other Collider

Class PlayerAttack

Namespace: [SamuraiSoccer.Player](#)

Assembly: SamuraiSoccer.Player.dll

```
public class PlayerAttack : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← PlayerAttack

Fields

slash

```
public GameObject slash
```

Field Value

GameObject

Methods

Attack()

```
private void Attack()
```

Start()

```
private void Start()
```

Class PlayerEffect

Namespace: [SamuraiSoccer.Player](#)

Assembly: SamuraiSoccer.Player.dll

```
public class PlayerEffect : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← PlayerEffect

Fields

aura

```
[SerializeField]  
private GameObject aura
```

Field Value

GameObject

m_isPlaying

```
private bool m_isPlaying
```

Field Value

bool

Methods

Start()

```
private void Start()
```

Class PlayerMove

Namespace: [SamuraiSoccer.Player](#)

Assembly: SamuraiSoccer.Player.dll

```
public class PlayerMove : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← PlayerMove

Fields

alpha

```
[SerializeField]  
private float alpha
```

Field Value

[float](#)

boundy

```
private Rect boundy
```

Field Value

Rect

currentSpeed

```
private float currentSpeed
```

Field Value

[float](#)

field

```
public FieldManager field
```

Field Value

[FieldManager](#)

flagsParent

```
public Transform flagsParent
```

Field Value

Transform

lastSlash

```
private DateTime lastSlash
```

Field Value

[DateTime](#)

m_state

```
private PlayerMove.State m_state
```

Field Value

rigidbody

```
private Rigidbody rigidbody
```

Field Value

Rigidbody

slashCollider

```
[SerializeField]  
private GameObject slashCollider
```

Field Value

GameObject

slashTrail

```
[SerializeField]  
private GameObject slashTrail
```

Field Value

GameObject

speed

```
[SerializeField]  
private float speed
```

Field Value

[float](#)

speedUpCoeff

```
[SerializeField]  
private float speedUpCoeff
```

Field Value

[float](#)

velocity

```
private Vector2 velocity
```

Field Value

Vector2

wakeUpT

```
[SerializeField]  
private float wakeUpT
```

Field Value

[float](#)

Methods

CalcRealVec(float, float, Vector2)

```
private Vector2 CalcRealVec(float x, float y, Vector2 currentVelocity)
```

Parameters

x [float](#)

y [float](#)

currentVelocity Vector2

Returns

Vector2

ChargeAttack()

ため攻撃

```
private UniTask ChargeAttack()
```

Returns

UniTask

CheckBoundy(Vector3, Vector2)

```
private Vector2 CheckBoundy(Vector3 pos, Vector2 dir)
```

Parameters

pos Vector3

dir Vector2

Returns

Vector2

Move(Vector3)

```
private void Move(Vector3 stickDir)
```

Parameters

stickDir Vector3

ReceiveStick(Vector3)

```
private void ReceiveStick(Vector3 stickDir)
```

Parameters

stickDir Vector3

SetBoundy()

コーナーフラグの位置を境界に設定

```
private Rect SetBoundy()
```

Returns

Rect

Start()

```
private void Start()
```

UpdateCurrentSpeed()

```
private void UpdateCurrentSpeed()
```

UpdateSlashTime()

```
private void UpdateSlashTime()
```

Enum PlayerMove.State

Namespace: [SamuraiSoccer.Player](#)

Assembly: SamuraiSoccer.Player.dll

```
private enum PlayerMove.State
```

Fields

ChargeAttack = 2

Idle = 3

Playing = 1

StandBy = 0

Class Slash

Namespace: [SamuraiSoccer.Player](#)

Assembly: SamuraiSoccer.Player.dll

```
public class Slash : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← Slash

Fields

alpha

```
private float alpha
```

Field Value

[float](#)

animator

```
public Animator animator
```

Field Value

Animator

m_isAttackBall

```
private bool m_isAttackBall
```

Field Value

[bool](#)

particle

```
private ParticleSystem.MainModule particle
```

Field Value

ParticleSystem.MainModule

slash

```
public AudioClip slash
```

Field Value

AudioClip

time

```
private float time
```

Field Value

[float](#)

Methods

OnHit(GameObject)

```
private void OnHit(GameObject obj)
```

Parameters

obj GameObject

Start()

```
private void Start()
```

Update()

```
private void Update()
```

Class TESTCHARGEATTACK

Namespace: [SamuraiSoccer.Player](#)

Assembly: SamuraiSoccer.Player.dll

```
public class TESTCHARGEATTACK : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← TESTCHARGEATTACK

Methods

OnClick()

```
public void OnClick()
```

Start()

```
private void Start()
```

Update()

```
private void Update()
```

Class Trail

Namespace: [SamuraiSoccer.Player](#)

Assembly: SamuraiSoccer.Player.dll

```
public class Trail : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← Trail

Methods

OnHit(GameObject)

```
private void OnHit(GameObject obj)
```

Parameters

obj GameObject

Start()

```
private void Start()
```

Vanish(CancellationToken)

```
private UniTask Vanish(CancellationToken token)
```

Parameters

token [CancellationToken](#)

Returns

Namespace SamuraiSoccer.SoccerGame

Classes

[BallAction](#)

ボールに対するアクションをするクラス。 (ボールにくっつける。)

[BallActionCommand](#)

ボールを操作するコマンドの元。

[BallSlashed](#)

[Constants](#)

定数

[DribbleCommand](#)

ドリブルをする時のコマンド。

[Extentions](#)

[FieldBoundary](#)

[FieldInfo](#)

フィールド

[FieldManager](#)

フィールド情報を管理する。

[FieldRotationBase](#)

フィールドの傾きを持ったクラス。ここから派生。

[GameProcess](#)

[NonRotation](#)

[NonWind](#)

[PassCommand](#)

パスをする時のコマンド。

[Penalty](#)

[PersonalStatus](#)

[RefereeArea](#)

[RefereeMove](#)

[ScoreManager](#)

得点した際の処理を管理するクラス。

[ShootCommand](#)

シュートする時のコマンド。

[StagePrefabManager](#)

[TrapCommand](#)

トラップする時のコマンド。

[TutorialPenalty](#)

[WindInfoBase](#)

風の情報を持ったクラス。これから派生させてください。

Interfaces

[ISlashed](#)

Enums

[PassHeight](#)

パスの高さ。

[RefereeMove.State](#)

Class BallAction

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

ボールに対するアクションをするクラス。 (ボールにくつける。)

```
public class BallAction : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← BallAction

Fields

angularVelocity

```
private Vector3 angularVelocity
```

Field Value

Vector3

calledNum

```
private int calledNum
```

Field Value

[int](#)

commandStream

```
private static Subject<BallActionCommand> commandStream
```

Field Value

Subject<[BallActionCommand](#)>

gravity

```
private static readonly float gravity
```

Field Value

[float](#)

info

```
public FieldManager info
```

Field Value

[FieldManager](#)

isPause

```
private bool isPause
```

Field Value

[bool](#)

last_touch

```
[NonSerialized]  
public bool last_touch
```

Field Value

[bool](#)

Owner

[NonSerialized]

public GameObject owner

Field Value

GameObject

rb

[NonSerialized]

public Rigidbody rb

Field Value

Rigidbody

scoreManager

public ScoreManager scoreManager

Field Value

[ScoreManager](#)

sqrt2

private static readonly float sqrt2

Field Value

[float](#) ↗

sqrt3

```
private static readonly float sqrt3
```

Field Value

[float](#) ↗

velocity

```
private Vector3 velocity
```

Field Value

Vector3

Methods

Awake()

```
private void Awake()
```

CalcHighPass(Vector2, Vector2, PersonalStatus)

```
private void CalcHighPass(Vector2 sender, Vector2 receiver, PersonalStatus self)
```

Parameters

sender Vector2

receiver Vector2

self [PersonalStatus](#)

CalcLowPass(Vector2, Vector2, PersonalStatus)

```
private void CalcLowPass(Vector2 sender, Vector2 receiver, PersonalStatus self)
```

Parameters

sender Vector2

receiver Vector2

self [PersonalStatus](#)

CalcMiddlePass(Vector2, Vector2, PersonalStatus)

```
private void CalcMiddlePass(Vector2 sender, Vector2 receiver, PersonalStatus self)
```

Parameters

sender Vector2

receiver Vector2

self [PersonalStatus](#)

Command(BallActionCommand)

コマンドの処理を振り分け。

```
private void Command(BallActionCommand command)
```

Parameters

command [BallActionCommand](#)

コマンド。

CommandStreamOnNext(BallActionCommand)

ボールに対するアクションを行う。

```
public static void CommandStreamOnNext(BallActionCommand command)
```

Parameters

command [BallActionCommand](#)

コマンド。

Dribble(DribbleCommand)

ドリブルっぽいもの。StartCoroutineじゃなくてイテレーターで操作してほしい。

```
public UniTask Dribble(DribbleCommand c)
```

Parameters

c [DribbleCommand](#)

Returns

UniTask

OnCollisionEnter(Collision)

特殊なイベント発生

```
private void OnCollisionEnter(Collision other)
```

Parameters

other Collision

OnTriggerEnter(Collider)

```
private void OnTriggerEnter(Collider other)
```

Parameters

other Collider

Pass(PassCommand)

パスをするとき呼ぶ関数

```
private void Pass(PassCommand command)
```

Parameters

command PassCommand

Pause(bool)

```
private void Pause(bool isPause)
```

Parameters

isPause bool ↗

Play(Unit)

```
private void Play(Unit _)
```

Parameters

_ Unit

Shoot(ShootCommand)

ショットを撃つ関数

```
private void Shoot(ShootCommand command)
```

Parameters

command [ShootCommand](#)

Start()

```
private void Start()
```

Steal(GameObject, PersonalStatus, PersonalStatus)

ボールを盗むときの関数

```
private void Steal(GameObject self, PersonalStatus holder = null, PersonalStatus tryer = null)
```

Parameters

self GameObject

ボール奪う人

holder [PersonalStatus](#)

ボールを持っている人の能力値

tryer [PersonalStatus](#)

ボールを奪う人の能力値

SuccessSteal(PersonalStatus, PersonalStatus)

```
private bool SuccessSteal(PersonalStatus holder, PersonalStatus tryer)
```

Parameters

holder [PersonalStatus](#)

tryer [PersonalStatus](#)

Returns

[bool](#) ↗

SuccessTrap(PersonalStatus)

トラップが成功するかの判定

```
private bool SuccessTrap(PersonalStatus self)
```

Parameters

self [PersonalStatus](#)

能力値

Returns

[bool](#) ↗

成功のときtrue

Trap(TrapCommand)

トラップするとき呼ぶ関数

```
private void Trap(TrapCommand command)
```

Parameters

command [TrapCommand](#)

Update()

`private void Update()`

Class BallActionCommand

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

ボールを操作するコマンドの元。

```
public abstract class BallActionCommand
```

Inheritance

[object](#) ↗ ← BallActionCommand

Derived

[DribbleCommand](#), [PassCommand](#), [ShootCommand](#), [TrapCommand](#)

Class BallSlashed

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

```
public class BallSlashed : MonoBehaviour, ISlashed
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← BallSlashed

Implements

[ISlashed](#)

Fields

m_power

```
[SerializeField]  
private float m_power
```

Field Value

[float](#)

m_rigidbody

```
[SerializeField]  
private Rigidbody m_rigidbody
```

Field Value

Rigidbody

Methods

Slashed(Vector3)

衝撃波に斬られる時に呼ばれる

```
public void Slashed(Vector3 dir)
```

Parameters

dir Vector3

吹っ飛ぶ方向

Class Constants

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

定数

```
public static class Constants
```

Inheritance

[object](#) ↗ ← Constants

Fields

OppormentGoalPoint

相手のゴール

```
public static readonly Vector3 OppormentGoalPoint
```

Field Value

Vector3

OppormentSpornRandMaxX

```
public static readonly float OppormentSpornRandMaxX
```

Field Value

[float](#) ↗

OppormentSpornRandMaxZ

```
public static readonly float OppornentSpornRandMaxZ
```

Field Value

[float](#)

OppornentSpornRandMinX

相手がスปーンするランダム幅

```
public static readonly float OppornentSpornRandMinX
```

Field Value

[float](#)

OppornentSpornRandMinZ

```
public static readonly float OppornentSpornRandMinZ
```

Field Value

[float](#)

OurGoalPoint

自陣のゴール

```
public static readonly Vector3 OurGoalPoint
```

Field Value

Vector3

opponentInitialSpornPointCenterOpponent

```
private static readonly IEnumerable<Vector3> opponentInitialSpornPointCenterOpponent
```

Field Value

[IEnumerable](#)<Vector3>

opponentInitialSpornPointCenterTeam

```
private static readonly IEnumerable<Vector3> opponentInitialSpornPointCenterTeam
```

Field Value

[IEnumerable](#)<Vector3>

teammateInitialSpornPointCenterOpponent

```
private static readonly IEnumerable<Vector3> teammateInitialSpornPointCenterOpponent
```

Field Value

[IEnumerable](#)<Vector3>

teammateInitialSpornPointCenterTeam

```
private static readonly IEnumerable<Vector3> teammateInitialSpornPointCenterTeam
```

Field Value

[IEnumerable](#)<Vector3>

Properties

G2G

ゴール間の距離

```
public static float G2G { get; }
```

Property Value

[float](#)

OppornentSpornPoint

相手のスปーン地点

```
public static Vector3 OppornentSpornPoint { get; }
```

Property Value

Vector3

OpprnentInitialSpornPointCenterOppornt

相手ボールのときの相手の最初のスปーン場所

```
public static Vector3[] OpprnentInitialSpornPointCenterOppornt { get; }
```

Property Value

Vector3[]

OpprnentInitialSpornPointCenterTeam

味方ボールのときの相手の最初のスปーン場所

```
public static Vector3[] OpprnentInitialSpornPointCenterTeam { get; }
```

Property Value

Vector3[]

TeammateInitialSpornPointCenterOpporment

相手ボールのときの味方の最初のスปーン場所

```
public static Vector3[] TeammateInitialSpornPointCenterOpporment { get; }
```

Property Value

Vector3[]

TeammateInitialSpornPointCenterTeam

味方ボールのときの味方の最初のスปーン場所

```
public static Vector3[] TeammateInitialSpornPointCenterTeam { get; }
```

Property Value

Vector3[]

TeammateSpornPoint

味方のスปーン地点

```
public static Vector3 TeammateSpornPoint { get; }
```

Property Value

Vector3

Width

フィールドの幅

```
public static float Width { get; }
```

Property Value

[float↗](#)

Class DribbleCommand

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

ドリブルをする時のコマンド。

```
public class DribbleCommand : BallActionCommand
```

Inheritance

[object](#) ← [BallActionCommand](#) ← DribbleCommand

Constructors

DribbleCommand(Transform, PersonalStatus)

ドリブルをする時のコマンド。

```
public DribbleCommand(Transform owner, PersonalStatus status)
```

Parameters

owner Transform

ドリブル主。

status [PersonalStatus](#)

ドリブル主のステータス。

Fields

m_owner

```
public Transform m_owner
```

Field Value

Transform

m_status

```
public PersonalStatus m_status
```

Field Value

[PersonalStatus](#)

Class Extentions

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

```
public static class Extentions
```

Inheritance

[object](#) ← Extentions

Methods

ToVector2(Transform)

```
public static Vector2 ToVector2(this Transform t)
```

Parameters

t Transform

Returns

Vector2

ToVector2Int(GameObject)

```
public static Vector2Int ToVector2Int(this GameObject obj)
```

Parameters

obj GameObject

Returns

Vector2Int

Class FieldBoundary

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

```
public class FieldBoundary : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← FieldBoundary

Fields

m_maxFlag

```
[SerializeField]  
private GameObject m_maxFlag
```

Field Value

GameObject

m_minFlag

```
[SerializeField]  
private GameObject m_minFlag
```

Field Value

GameObject

m_xMax

```
private static float m_xMax
```

Field Value

[float](#) ↗

m_xMin

```
private static float m_xMin
```

Field Value

[float](#) ↗

m_zMax

```
private static float m_zMax
```

Field Value

[float](#) ↗

m_zMin

```
private static float m_zMin
```

Field Value

[float](#) ↗

Properties

XMax

```
public static float XMax { get; }
```

Property Value

[float](#) ↗

XMin

```
public static float XMin { get; }
```

Property Value

[float](#) ↗

ZMax

```
public static float ZMax { get; }
```

Property Value

[float](#) ↗

ZMin

```
public static float ZMin { get; }
```

Property Value

[float](#) ↗

Methods

Start()

```
private void Start()
```

Class FieldInfo

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

フィールド

```
[Serializable]
public class FieldInfo
```

Inheritance

[object](#) ← FieldInfo

Fields

acc_down_coeff

```
public float[][] acc_down_coeff
```

Field Value

[float](#)[][]

acc_up_coeff

```
public float[][] acc_up_coeff
```

Field Value

[float](#)[][]

drag

```
public float[][] drag
```

Field Value

[float](#) [][]

Methods

GetAccDownCoeff(Vector3)

```
public float GetAccDownCoeff(Vector3 position)
```

Parameters

position Vector3

Returns

[float](#) []

GetAccUpCoeff(Vector3)

```
public float GetAccUpCoeff(Vector3 position)
```

Parameters

position Vector3

Returns

[float](#) []

Getdrag(Vector3)

```
public float Getdrag(Vector3 position)
```

Parameters

position Vector3

Returns

[float](#) ↗

Class FieldManager

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

フィールド情報を管理する。

```
[DefaultExecutionOrder(-1)]  
public class FieldManager : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← FieldManager

Fields

ball

```
public GameObject ball
```

Field Value

GameObject

ball_rb

```
private Rigidbody ball_rb
```

Field Value

Rigidbody

groundColor

```
private int groundNumber
```

Field Value

[int↗](#)

info

```
[NonSerialized]  
public FieldInfo info
```

Field Value

[FieldInfo](#)

root

```
public GameObject root
```

Field Value

GameObject

rotation

```
public FieldRotationBase rotation
```

Field Value

[FieldRotationBase](#)

wind

```
private WindInfoBase wind
```

Field Value

[WindInfoBase](#)

Properties

GroundNumber

```
public int GroundNumber { get; }
```

Property Value

[int↗](#)

Methods

AdaptInversePosition(Vector3)

開店前の位置に変換

```
public Vector3 AdaptInversePosition(Vector3 pos)
```

Parameters

pos Vector3

元の場所

Returns

Vector3

変換後の場所

AdaptPosition(Vector3)

回転後の位置に変換

```
public Vector3 AdaptPosition(Vector3 pos)
```

Parameters

pos Vector3

元の場所

Returns

Vector3

変換後の場所

Awake()

```
private void Awake()
```

LoadField()

```
private UniTask LoadField()
```

Returns

UniTask

Update()

```
private void Update()
```

Class FieldRotationBase

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

フィールドの傾きを持ったクラス。ここから派生。

```
public abstract class FieldRotationBase : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← FieldRotationBase

Derived

[NonRotation](#)

Fields

rotation

回転

```
public Quaternion rotation
```

Field Value

Quaternion

Methods

AdaptInversePosition(Vector3)

```
public Vector3 AdaptInversePosition(Vector3 pos)
```

Parameters

pos Vector3

Returns

Vector3

AdaptPosition(Vector3)

```
public Vector3 AdaptPosition(Vector3 pos)
```

Parameters

pos Vector3

Returns

Vector3

Awake()

初期化。ご自由に。

```
protected virtual void Awake()
```

Class GameProcess

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

```
public class GameProcess : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← GameProcess

Fields

m_gameBGM

```
[SerializeField]  
[Tooltip("ゲームのBGM、 -1の時はなにも流さない")]  
private int m_gameBGM
```

Field Value

[int](#)

m_trumpetShellSENumber

```
[SerializeField]  
[Tooltip("ほら貝のSE、 -1の時はなにも流さない")]  
private int m_trumpetShellSENumber
```

Field Value

[int](#)

m_whistleSENumber

```
[SerializeField]
[Tooltip("ホイッスル開始音、-1の時はなにも流さない")]
private int m_whistleSENumber
```

Field Value

[int↗](#)

Methods

Awake()

```
private void Awake()
```

FirstStandbyContents()

```
private UniTask FirstStandbyContents()
```

Returns

UniTask

ResetContents()

```
private UniTask ResetContents()
```

Returns

UniTask

Start()

```
private void Start()
```

Interface ISlashed

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

```
public interface ISlashed
```

Methods

Slashed(Vector3)

衝撃波に斬られる時に呼ばれる

```
void Slashed(Vector3 dir)
```

Parameters

dir Vector3

吹っ飛ぶ方向

Class NonRotation

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

```
public class NonRotation : FieldRotationBase
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← [FieldRotationBase](#) ← NonRotation

Inherited Members

[FieldRotationBase.rotation](#) , [FieldRotationBase.AdaptPosition\(Vector3\)](#) ,
[FieldRotationBase.AdaptInversePosition\(Vector3\)](#).

Methods

Awake()

初期化。ご自由に。

```
protected override void Awake()
```

Class NonWind

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

```
public class NonWind : WindInfoBase
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← [WindInfoBase](#) ← NonWind

Inherited Members

[WindInfoBase.wind](#) , [WindInfoBase.WindEffect\(Vector3\)](#)

Methods

Start()

初期化。派生クラスでは自由に

```
protected override void Start()
```

Class PassCommand

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

パスをする時のコマンド。

```
public class PassCommand : BallActionCommand
```

Inheritance

[object](#) ← [BallActionCommand](#) ← PassCommand

Constructors

PassCommand(Vector2, Vector2, PassHeight, PersonalStatus)

パスをする時のコマンド。

```
public PassCommand(Vector2 sender, Vector2 receiver, PassHeight passHeight,  
PersonalStatus status)
```

Parameters

sender Vector2

パス主。

receiver Vector2

受け取り主。

passHeight [PassHeight](#)

パスの高さ。

status [PersonalStatus](#)

パス主のステータス。

Fields

m_passHeight

```
public PassHeight m_passHeight
```

Field Value

[PassHeight](#)

m_receiver

```
public Vector2 m_receiver
```

Field Value

Vector2

m_sender

```
public Vector2 m_sender
```

Field Value

Vector2

m_status

```
public PersonalStatus m_status
```

Field Value

[PersonalStatus](#)

Enum PassHeight

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

パスの高さ.

```
public enum PassHeight
```

Fields

High = 2

Low = 0

Middle = 1

Class Penalty

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

```
public class Penalty : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← Penalty

Fields

m_gameOverPanel

```
[SerializeField]  
private GameObject m_gameOverPanel
```

Field Value

GameObject

m_redcardSENumber

```
private int m_redcardSENumber
```

Field Value

[int](#)

m_resultSceneName

```
[SerializeField]  
private string m_resultSceneName
```

Field Value

[string](#) ↗

m_yellowCard

```
[SerializeField]  
private GameObject[] m_yellowCard
```

Field Value

GameObject[]

m_yellowcardSENumber

```
private int m_yellowcardSENumber
```

Field Value

[int](#) ↗

Methods

LoseEffect()

```
private UniTask LoseEffect()
```

Returns

UniTask

OnReset(Unit)

```
private void OnReset(Unit _)
```

Parameters

— Unit

Start()

```
private void Start()
```

YellowCard(int)

```
private void YellowCard(int penaltynumber)
```

Parameters

penaltynumber [int ↗](#)

Class PersonalStatus

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

```
[Serializable]
public class PersonalStatus
```

Inheritance

[object](#) ← PersonalStatus

Fields

MAX_HP

```
public int MAX_HP
```

Field Value

[int](#)

ally

```
public bool ally
```

Field Value

[bool](#)

fast

```
public float fast
```

Field Value

[float](#) ↗

hp

`public int hp`

Field Value

[int](#) ↗

power

`public float power`

Field Value

[float](#) ↗

seelen

`public float seelen`

Field Value

[float](#) ↗

Class RefereeArea

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

```
public class RefereeArea : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← RefereeArea

Fields

m_areaSize

```
[Tooltip("審判の視界の距離")]
private float m_areaSize
```

Field Value

[float](#)

m_isPlaying

```
private IReadOnlyReactiveProperty<bool> m_isPlaying
```

Field Value

IReadOnlyReactiveProperty<[bool](#)>

m_maxang

```
[Tooltip("審判の視界の視野角")]
private float m_maxang
```

Field Value

[float](#) ↗

m_meshFilter

`private MeshFilter m_meshFilter`

Field Value

MeshFilter

m_penaltyCount

`private int m_penaltyCount`

Field Value

[int](#) ↗

m_player

`private Transform m_player`

Field Value

Transform

surprisedMark

`public ParticleSystem surprisedMark`

Field Value

useObstacles

```
[Space(10)]  
public bool useObstacles
```

Field Value

[bool](#) ↗

Properties

AreaSize

```
public float AreaSize { get; }
```

Property Value

[float](#) ↗

MaxAngle

```
public float MaxAngle { get; }
```

Property Value

[float](#) ↗

Methods

Check()

```
private bool Check()
```

Returns

[bool](#)

DynamicMeshMaker()

```
private void DynamicMeshMaker()
```

FoulCheck(CancellationToken)

```
private UniTask FoulCheck(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

Returns

UniTask

MeshMaker()

```
public void MeshMaker()
```

SetAreaSize(float)

```
public void SetAreaSize(float newValue)
```

Parameters

`newValue float`

SetMaxAngle(float)

```
public void SetMaxAngle(float newValue)
```

Parameters

`newValue float`

Start()

```
private void Start()
```

Update()

```
private void Update()
```

Class RefereeMove

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

```
public class RefereeMove : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← RefereeMove

Derived

[KunfuReferee](#)

Fields

anicon

```
public Animator anicon
```

Field Value

Animator

ball

```
public GameObject ball
```

Field Value

GameObject

lookatspeed

```
public float lookatspeed
```

Field Value

[float](#)

radius

```
public float radius
```

Field Value

[float](#)

refereeArea

```
public RefereeArea refereeArea
```

Field Value

[RefereeArea](#)

rig

```
private Rigidbody rig
```

Field Value

Rigidbody

runningspeed

```
public float runningspeed
```

Field Value

[float](#)

state

```
private RefereeMove.State state
```

Field Value

[RefereeMove.State](#)

Methods

LookAtBall()

```
protected virtual void LookAtBall()
```

MoveAroundBall()

```
private void MoveAroundBall()
```

Start()

```
protected virtual void Start()
```

Update()

```
protected virtual void Update()
```

Enum RefereeMove.State

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

```
private enum RefereeMove.State
```

Fields

Moving = 0

Stop = 1

Class ScoreManager

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

得点した際の処理を管理するクラス。

```
public class ScoreManager : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← ScoreManager

Fields

m_CutSceneScoreOpponent

```
[SerializeField]  
private List<int> m_CutSceneScoreOpponent
```

Field Value

[List](#)<[int](#)>

m_CutSceneScoreTeammate

```
[SerializeField]  
private List<int> m_CutSceneScoreTeammate
```

Field Value

[List](#)<[int](#)>

m_opponentScore

```
private int m_opponentScore
```

Field Value

[int](#)

m_teammateScore

```
private int m_teammateScore
```

Field Value

[int](#)

Methods

Goal(bool)

ゴール時に呼ぶ。

```
public void Goal(bool isTeammate)
```

Parameters

isTeammate [bool](#)

自チームの得点か。

Class ShootCommand

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

シュートする時のコマンド。

```
public class ShootCommand : BallActionCommand
```

Inheritance

[object](#) ↗ ← [BallActionCommand](#) ← ShootCommand

Constructors

ShootCommand(Transform, PersonalStatus)

シュートする時のコマンド。

```
public ShootCommand(Transform sender, PersonalStatus status)
```

Parameters

sender Transform

シュート主。

status [PersonalStatus](#)

シュート主のステータス。

Fields

m_sender

```
public Transform m_sender
```

Field Value

Transform

m_status

```
public PersonalStatus m_status
```

Field Value

[PersonalStatus](#)

Class StagePrefabManager

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

```
public class StagePrefabManager : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← StagePrefabManager

Fields

crowdMaterial

```
public Material crowdMaterial
```

Field Value

Material

crowdRenderers

```
public Renderer[] crowdRenderers
```

Field Value

Renderer[]

crowdTextures

```
public Texture[] crowdTextures
```

Field Value

Texture[]

flagRenderers

```
public MeshRenderer[] flagRenderers
```

Field Value

MeshRenderer[]

flagShader

```
public Shader flagShader
```

Field Value

Shader

flagTexture

```
public Texture flagTexture
```

Field Value

Texture

groundMaterial

```
public Material groundMaterial
```

Field Value

Material

groundNormalMap

```
public Texture groundNormalMap
```

Field Value

Texture

groundRenderer

```
public Renderer groundRenderer
```

Field Value

Renderer

groundTexture

```
public Texture groundTexture
```

Field Value

Texture

lookAtSpeed

```
public float lookAtSpeed
```

Field Value

[float](#)

refereeArea

```
public RefereeArea refereeArea
```

Field Value

[RefereeArea](#)

refereeAreaSize

```
public int refereeAreaSize
```

Field Value

[int](#)

refereeMaxAng

```
public int refereeMaxAng
```

Field Value

[int](#)

refereeMove

```
public RefereeMove refereeMove
```

Field Value

[RefereeMove](#)

runningSpeed

```
public float runningSpeed
```

Field Value

[float](#)

useObstacles

```
public bool useObstacles
```

Field Value

[bool](#)

Methods

Start()

```
private void Start()
```

Class TrapCommand

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

トラップする時のコマンド。

```
public class TrapCommand : BallActionCommand
```

Inheritance

[object](#) ← [BallActionCommand](#) ← TrapCommand

Constructors

TrapCommand(Transform, PersonalStatus)

トラップする時のコマンド。

```
public TrapCommand(Transform receiver, PersonalStatus status)
```

Parameters

receiver Transform

トラップ主。

status [PersonalStatus](#)

トラップ主のステータス。

Fields

m_receiver

```
public Transform m_receiver
```

Field Value

Transform

m_status

```
public PersonalStatus m_status
```

Field Value

[PersonalStatus](#)

Class TutorialPenalty

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

```
public class TutorialPenalty : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← TutorialPenalty

Fields

m_redcardSENumber

```
private int m_redcardSENumber
```

Field Value

[int](#)

m_yellowCard

```
[SerializeField]  
private GameObject[] m_yellowCard
```

Field Value

GameObject[]

m_yellowcardSENumber

```
private int m_yellowcardSENumber
```

Field Value

[int ↗](#)

Methods

OnReset(Unit)

```
private void OnReset(Unit _)
```

Parameters

_ Unit

Start()

```
private void Start()
```

YellowCard(int)

```
public void YellowCard(int penaltynumber)
```

Parameters

penaltynumber [int ↗](#)

Class WindInfoBase

Namespace: [SamuraiSoccer.SoccerGame](#)

Assembly: SamuraiSoccer.SoccerGame.dll

風の情報を持ったクラス。これから派生させてください。

```
public abstract class WindInfoBase : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← WindInfoBase

Derived

[NonWind](#)

Fields

wind

風の力

```
public Vector3[][] wind
```

Field Value

Vector3[][]

Methods

Start()

初期化。派生クラスでは自由に

```
protected virtual void Start()
```

WindEffect(Vector3)

```
public Vector3 WindEffect(Vector3 position)
```

Parameters

position Vector3

Returns

Vector3

Namespace SamuraiSoccer.SoccerGame.AI

Classes

[EasyCPU](#)

簡単なCPUの動作をさせるクラス。

[EasyCPUManager](#)

CPUを操作するクラス

[EnemySlashed](#)

[FatEnemySlashed](#)

[SwitchCPU](#)

[Team](#)

[TutorialEnemySlashed](#)

Structs

[SwitchCPU.WeightSet](#)

Class EasyCPU

Namespace: [SamuraiSoccer.SoccerGame.AI](#)

Assembly: SamuraiSoccer.SoccerGame.AI.dll

簡単なCPUの動作をさせるクラス。

```
public class EasyCPU : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← EasyCPU

Fields

ball

```
public Transform ball
```

Field Value

Transform

before_velocity

```
private Vector2 before_velocity
```

Field Value

Vector2

field

```
public FieldManager field
```

Field Value

[FieldManager](#)

m_isPause

```
private bool m_isPause
```

Field Value

[bool](#) ↗

manager

```
public EasyCPUManager manager
```

Field Value

[EasyCPUManager](#)

rb

```
public Rigidbody rb
```

Field Value

Rigidbody

rot_chain

```
private LinkedList<Vector2> rot_chain
```

Field Value

[LinkedList](#)<Vector2>

status

`public PersonalStatus status`

Field Value

[PersonalStatus](#)

Methods

AllMove(Vector2)

`private void AllMove(Vector2 dest)`

Parameters

`dest` Vector2

Attacked()

攻撃されたときに呼ぶ。hpが減り、吹っ飛びやすくなる。

`public void Attacked()`

CalcNextPoint(Vector2)

`private Vector2 CalcNextPoint(Vector2 vec)`

Parameters

`vec` Vector2

Returns

Vector2

CalcRealVec(Vector2)

```
private Vector2 CalcRealVec(Vector2 vec)
```

Parameters

vec Vector2

Returns

Vector2

CalcRotAverage()

```
private float CalcRotAverage()
```

Returns

float ↗

OnCollisionEnter(Collision)

```
private void OnCollisionEnter(Collision other)
```

Parameters

other Collision

Pause(bool)

```
private void Pause(bool isPause)
```

Parameters

isPause bool

Play(Unit)

```
private void Play(Unit _)
```

Parameters

_ Unit

SetMass()

吹っ飛びやすさを再設定

```
public void SetMass()
```

SetPause(bool)

ポーズ状態が設定する。初期化のとき呼ぶ。

```
public void SetPause(bool isPause)
```

Parameters

isPause bool

ポーズか。

SlowDown()

```
public void SlowDown()
```

SlowMove()

```
private UniTask SlowMove()
```

Returns

UniTask

Standby(bool)

```
private void Standby(bool _)
```

Parameters

_ [bool](#) ↗

Start()

```
private void Start()
```

Update()

```
private void Update()
```

Class EasyCPUManager

Namespace: [SamuraiSoccer.SoccerGame.AI](#)

Assembly: SamuraiSoccer.SoccerGame.AI.dll

CPUを操作するクラス

```
[DefaultExecutionOrder(5)]  
[RequireComponent(typeof(FieldManager))]  
public class EasyCPUManager : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← EasyCPUManager

Fields

audioSource

```
public AudioSource audioSource
```

Field Value

ball

```
public BallAction ball
```

Field Value

[BallAction](#)

field

```
private FieldManager field
```

Field Value

[FieldManager](#)

goalSound

```
public AudioClip goalSound
```

Field Value

AudioClip

m_isPause

```
private bool m_isPause
```

Field Value

[bool](#)

near_opp

```
[NonSerialized]  
public GameObject near_opp
```

Field Value

GameObject

near_team

```
[NonSerialized]  
public GameObject near_team
```

Field Value

GameObject

opp

```
public List<GameObject> opp
```

Field Value

[List](#) <GameObject>

oppName

```
private string oppName
```

Field Value

[string](#)

opp_p

```
public Transform opp_p
```

Field Value

Transform

opp_stock

```
public Team opp_stock
```

Field Value

[Team](#)

opponent

```
private GameObject opponent
```

Field Value

GameObject

rbs

```
public Dictionary<GameObject, Rigidbody> rbs
```

Field Value

[Dictionary](#)<GameObject, Rigidbody>

referee

```
public GameObject referee
```

Field Value

GameObject

resultSceneName

```
public string resultSceneName
```

Field Value

[string](#)

samurai

```
public GameObject samurai
```

Field Value

GameObject

startSound

```
public AudioClip startSound
```

Field Value

AudioClip

team

```
public List<GameObject> team
```

Field Value

[List](#)<GameObject>

team_p

```
public Transform team_p
```

Field Value

Transform

team_stock

```
public Team team_stock
```

Field Value

[Team](#)

teammate

```
private GameObject teammate
```

Field Value

GameObject

teammateName

```
private string teammateName
```

Field Value

[string](#)

Properties

OpponentMemberCount

敵の人数

```
public int OpponentMemberCount { get; }
```

Property Value

[int↗](#)

TeamMemberCount

味方の人数

```
public int TeamMemberCount { get; }
```

Property Value

[int↗](#)

Methods

Awake()

```
private void Awake()
```

Boost(bool, float, int)

ブースト

```
public void Boost(bool isTeam, float coeff, int finishTime = 0)
```

Parameters

isTeam [bool↗](#)

味方にブーストか

coeff [float](#)

倍率

finishTime [int](#)

ブースト終了時間

FinBoost(bool, float, int)

```
private UniTask FinBoost(bool isTeam, float coeff, int fin)
```

Parameters

isTeam [bool](#)

coeff [float](#)

fin [int](#)

Returns

UniTask

GoalAction(GoalEventType)

```
private UniTask GoalAction(GoalEventType t)
```

Parameters

t [GoalEventType](#)

Returns

UniTask

Init(bool)

初期化。選手の生成をしてる。

```
public void Init(bool centerIsTeammate = false)
```

Parameters

centerIsTeammate [bool](#)

Kill(GameObject)

選手を殺す。一応瞬時復活もさせる。

```
public UniTask Kill(GameObject dead)
```

Parameters

dead [GameObject](#)

死ぬ対象の選手

Returns

[UniTask](#)

LoadMember()

```
private UniTask LoadMember()
```

Returns

[UniTask](#)

Pause(bool)

```
private void Pause(bool isPause)
```

Parameters

`isPause` `bool` ↗

Play(Unit)

```
private void Play(Unit _)
```

Parameters

`_` `Unit`

SetAnimatorSpeed(float)

```
private void SetAnimatorSpeed(float speed)
```

Parameters

`speed` `float` ↗

SlowToWin()

勝った時にシーン遷移までゆっくりにして遷移させる

```
private UniTask SlowToWin()
```

Returns

`UniTask`

Sporn(PersonalStatus, Vector3)

選手復活

```
public GameObject Sporn(PersonalStatus status, Vector3 pos)
```

Parameters

status [PersonalStatus](#)

ステータス

pos Vector3

復活場所

Returns

GameObject

Standby(bool)

```
private void Standby(bool _)
```

Parameters

_ [bool](#)

Start()

```
private void Start()
```

Update()

```
private void Update()
```

Class EnemySlashed

Namespace: [SamuraiSoccer.SoccerGame.AI](#)

Assembly: SamuraiSoccer.SoccerGame.AI.dll

```
public class EnemySlashed : MonoBehaviour, ISlashed
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← EnemySlashed

Implements

[ISlashed](#)

Derived

[FatEnemySlashed](#)

Fields

m_easyCPU

```
public EasyCPU m_easyCPU
```

Field Value

[EasyCPU](#)

m_isKilled

```
[HideInInspector]  
public bool m_isKilled
```

Field Value

[bool](#)

m_rigidbody

```
public Rigidbody m_rigidbody
```

Field Value

Rigidbody

Properties

EasyCPUManager

```
public EasyCPUManager EasyCPUManager { get; set; }
```

Property Value

[EasyCPUManager](#)

Methods

Slashed(Vector3)

斬られたときの処理

```
public virtual void Slashed(Vector3 dir)
```

Parameters

dir Vector3

飛んでいく方向

Class FatEnemySlashed

Namespace: [SamuraiSoccer.SoccerGame.AI](#)

Assembly: SamuraiSoccer.SoccerGame.AI.dll

```
public class FatEnemySlashed : EnemySlashed, ISlashed
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← [EnemySlashed](#) ← FatEnemySlashed

Implements

[ISlashed](#)

Inherited Members

[EnemySlashed.m_easyCPU](#) , [EnemySlashed.m_rigidbody](#) , [EnemySlashed.EasyCPUManger](#) ,
[EnemySlashed.m_isKilled](#)

Methods

Slashed(Vector3)

斬られたときの処理

```
public override void Slashed(Vector3 dir)
```

Parameters

dir Vector3

飛んでいく方向

Class SwitchCPU

Namespace: [SamuraiSoccer.SoccerGame.AI](#)

Assembly: SamuraiSoccer.SoccerGame.AI.dll

```
public class SwitchCPU : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← SwitchCPU

Fields

gameObjectsAndWeight

```
public SwitchCPU.WeightSet[] gameObjectsAndWeight
```

Field Value

[WeightSet\[\]](#)

Methods

Start()

```
private void Start()
```

Struct SwitchCPU.WeightSet

Namespace: [SamuraiSoccer.SoccerGame.AI](#)

Assembly: SamuraiSoccer.SoccerGame.AI.dll

```
[Serializable]
public struct SwitchCPU.WeightSet
```

Inherited Members

[ValueType.Equals\(object\)](#) , [ValueType.GetHashCode\(\)](#) , [ValueType.ToString\(\)](#)

Fields

gameObject

```
public GameObject gameObject
```

Field Value

GameObject

weight

```
public float weight
```

Field Value

[float](#)

Class Team

Namespace: [SamuraiSoccer.SoccerGame.AI](#)

Assembly: SamuraiSoccer.SoccerGame.AI.dll

```
public class Team
```

Inheritance

[object](#) ← Team

Fields

member

```
public List<PersonalStatus> member
```

Field Value

[List](#) <[PersonalStatus](#)>

Class TutorialEnemySlashed

Namespace: [SamuraiSoccer.SoccerGame.AI](#)

Assembly: SamuraiSoccer.SoccerGame.AI.dll

```
public class TutorialEnemySlashed : MonoBehaviour, ISlashed
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← TutorialEnemySlashed

Implements

[ISlashed](#)

Fields

m_rigidbody

```
[SerializeField]  
private Rigidbody m_rigidbody
```

Field Value

Rigidbody

Methods

Slashed(Vector3)

衝撃波に斬られる時に呼ばれる

```
public void Slashed(Vector3 dir)
```

Parameters

dir Vector3

吹つ飛ぶ方向

Namespace **SamuraiSoccer.SoccerGame.Editor** **Exteision**

Classes

[StagePrefabManagerEditor](#)

Class StagePrefabManagerEditor

Namespace: [SamuraiSoccer.SoccerGame.EditorExtesion](#)

Assembly: SamuraiSoccer.SoccerGame.EditorExtension.dll

```
[CustomEditor(typeof(StagePrefabManager))]
public class StagePrefabManagerEditor : Editor, IPreviewable, IToolModeOwner
```

Inheritance

[object](#) ← Object ← ScriptableObject ← Editor ← StagePrefabManagerEditor

Implements

IPreviewable, IToolModeOwner

Inherited Members

Editor.HasLargeHeader() , [Editor.CreateEditorWithContext\(Object\[\], Object, Type\)](#) ,
Editor.CreateEditorWithContext(Object[], Object) ,
[Editor.CreateCachedEditorWithContext\(Object, Object, Type, ref Editor\)](#) ,
[Editor.CreateCachedEditorWithContext\(Object\[\], Object, Type, ref Editor\)](#) ,
[Editor.CreateCachedEditor\(Object, Type, ref Editor\)](#) ,
[Editor.CreateCachedEditor\(Object\[\], Type, ref Editor\)](#) , Editor.CreateEditor(Object) ,
[Editor.CreateEditor\(Object, Type\)](#) , Editor.CreateEditor(Object[]) , [Editor.CreateEditor\(Object\[\], Type\)](#) ,
Editor.GetSerializedObjectInternal() , Editor.InternalSetTargets(Object[]) ,
Editor.IToolModeOwner.GetWorldBoundsOfTargets() , Editor.GetWorldBoundsOfTarget(Object) ,
[Editor.IToolModeOwner.ModeSurvivesSelectionChange\(int\)](#) , Editor.OnForceReloadInspector() ,
[Editor.GetOptimizedGUIBlock\(bool, bool, out float\)](#) , Editor.OnOptimizedInspectorGUI(Rect) ,
[Editor.DrawPropertiesExcluding\(SerializedObject, params string\[\]\)](#) , Editor.DrawDefaultInspector() ,
Editor.Repaint() , Editor.CreateInspectorGUI() , Editor.RequiresConstantRepaint() , Editor.DrawHeader() ,
Editor.OnHeaderGUI() , Editor.OnHeaderControlsGUI() , Editor.ShouldHideOpenButton() ,
Editor.CanOpenMultipleObjects() , Editor.ShouldTryToMakeEditableOnOpen() ,
Editor.OnHeaderIconGUI(Rect) , [Editor.OnHeaderTitleGUI\(Rect, string\)](#) ,
Editor.DrawHeaderHelpAndSettingsGUI(Rect) , Editor.DrawFoldoutInspector(Object, ref Editor) ,
Editor.HasPreviewGUI() , Editor.GetPreviewTitle() , [Editor.RenderStaticPreview\(string, Object\[\], int, int\)](#) ,
Editor.OnPreviewGUI(Rect, GUIStyle) , Editor.OnInteractivePreviewGUI(Rect, GUIStyle) ,
Editor.OnPreviewSettings() , Editor.GetInfoString() , Editor.DrawPreview(Rect) ,
Editor.ReloadPreviewInstances() , Editor.IsEnabled() , Editor.UseDefaultMargins() ,
Editor.MoveNextTarget() , Editor.ResetTarget() , Editor.OnAssetStoreInspectorGUI() ,
Editor.IToolModeOwner.GetInstanceID() , Editor.preview , Editor.IToolModeOwner.areToolModesAvailable ,

`Editor.target` , `Editor.targets` , `Editor.referenceTargetIndex` , `Editor.targetTitle` , `Editor.serializedObject` ,
`Editor.finishedDefaultHeaderGUI`

Fields

`_isOpen`

```
private bool _isOpen
```

Field Value

[bool](#)

`_num`

```
private int _num
```

Field Value

[int](#)

`_target`

```
private StagePrefabManager _target
```

Field Value

[StagePrefabManager](#)

`_textures`

```
private Texture[] _textures
```

Field Value

Texture[]

crowd

```
private bool crowd
```

Field Value

[bool](#) ↗

useNormalMap

```
private bool useNormalMap
```

Field Value

[bool](#) ↗

Methods

Awake()

```
private void Awake()
```

OnInspectorGUI()

Implement this function to make a custom inspector.

```
public override void OnInspectorGUI()
```

Namespace SamuraiSoccer.StageContents

Classes

[CrowdMember](#)

[Flag](#)

[GroundTexture](#)

Enums

[GameResult](#)

Class CrowdMember

Namespace: [SamuraiSoccer.StageContents](#)

Assembly: SamuraiSoccer.StageContents.dll

```
public class CrowdMember : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← CrowdMember

Fields

m_crowdMaterial

```
[SerializeField]  
private Material m_crowdMaterial
```

Field Value

Material

m_crowdRenderers

```
[SerializeField]  
private Renderer[] m_crowdRenderers
```

Field Value

Renderer[]

m_crowdTextures

```
[SerializeField]  
private Texture[] m_crowdTextures
```

Field Value

Texture[]

Methods

Start()

```
private void Start()
```

Class Flag

Namespace: [SamuraiSoccer.StageContents](#)

Assembly: SamuraiSoccer.StageContents.dll

```
public class Flag : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← Flag

Fields

m_meshes

```
[SerializeField]  
private MeshRenderer[] m_meshes
```

Field Value

MeshRenderer[]

m_shader

```
[SerializeField]  
private Shader m_shader
```

Field Value

Shader

m_texture

```
[SerializeField]  
private Texture m_texture
```

Field Value

Texture

Methods

Start()

```
private void Start()
```

Enum GameResult

Namespace: [SamuraiSoccer.StageContents](#)

Assembly: SamuraiSoccer.StageContents.dll

```
public enum GameResult
```

Fields

Draw = 2

Lose = 1

Undefined = 3

Win = 0

Class GroundTexture

Namespace: [SamuraiSoccer.StageContents](#)

Assembly: SamuraiSoccer.StageContents.dll

```
public class GroundTexture : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← GroundTexture

Fields

m_groundNormalMap

```
[SerializeField]  
private Texture m_groundNormalMap
```

Field Value

Texture

m_groundTexture

```
[SerializeField]  
private Texture m_groundTexture
```

Field Value

Texture

Methods

Start()

```
private void Start()
```

Namespace SamuraiSoccer.StageContents.BattleDome

Classes

[BGMDData](#)

[BSA](#)

BSAファイルを扱うクラス。

[Indicator](#)

Class BGMDATA

Namespace: [SamuraiSoccer.StageContents.BattleDome](#)

Assembly: BattleDome.dll

```
[Serializable]
[JsonObject]
public class BGMDATA
```

Inheritance

[object](#) ← BGMDATA

Fields

data

```
[JsonProperty]
public List<List<float>> data
```

Field Value

[List](#) <[List](#) <[float](#)>>

f_table

```
[JsonProperty]
public List<float> f_table
```

Field Value

[List](#) <[float](#)>

time_span

```
[JsonProperty]  
public float time_span
```

Field Value

[float](#) ↗

Class BSA

Namespace: [SamuraiSoccer.StageContents.BattleDome](#)

Assembly: BattleDome.dll

BSAファイルを扱うクラス。

```
public sealed class BSA
```

Inheritance

[object](#) ↗ ← BSA

Fields

m_fCount

```
private byte m_fCount
```

Field Value

[byte](#) ↗

m_levelCount

```
private byte m_levelCount
```

Field Value

[byte](#) ↗

m_raw

```
private byte[] m_raw
```

Field Value

[byte](#)[]

m_sampleCount

`private uint m_sampleCount`

Field Value

[uint](#)

m_strideTime

`private float m_strideTime`

Field Value

[float](#)

Properties

FCount

周波数サンプル数。

`public byte FCount { get; private set; }`

Property Value

[byte](#)[]

IsAvailable

ファイルが読み込まれ使用できるかどうか。

```
public bool IsAvailable { get; private set; }
```

Property Value

[bool](#)

LevelCount

音圧レベルの分解能。

```
public byte LevelCount { get; private set; }
```

Property Value

[byte](#)

LevelUnit

音圧レベルを0-1にする時の正規化係数。

```
public double LevelUnit { get; }
```

Property Value

[double](#)

SampleCount

サンプル数。

```
public uint SampleCount { get; private set; }
```

Property Value

[uint](#)

StrideTime

サンプル間の時間間隔 (s)。

```
public float StrideTime { get; private set; }
```

Property Value

[float](#)

Methods

AvailableCheck()

ファイルが読み込まれて使用できるかチェック。

```
private void AvailableCheck()
```

Exceptions

[MemberAccessException](#)

ファイルが読み込まれていない場合投げられる。

LevelData(int)

音圧レベルのデータを取得する。

```
public ReadOnlySpan<byte> LevelData(int time)
```

Parameters

time [int](#)

時間のインデックス。

Returns

[ReadOnlySpan](#) <[byte](#)>

音圧レベルのデータ。

Exceptions

[IndexOutOfRangeException](#)

インデックスが不正の場合投げられる。

Load(string)

ファイルをロードする。

```
public UniTask Load(string fileName)
```

Parameters

fileName [string](#)

読み込むファイル名。

Returns

UniTask

UniTask

Class Indicator

Namespace: [SamuraiSoccer.StageContents.BattleDome](#)

Assembly: BattleDome.dll

```
public class Indicator : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← Indicator

Fields

bgmData

```
private BSA bgmData
```

Field Value

[BSA](#)

m_bgm

```
public AudioSource m_bgm
```

Field Value

m_center

```
private readonly Vector3 m_center
```

Field Value

Vector3

m_fileName

```
private readonly string m_fileName
```

Field Value

[string](#) ↗

m_indicatorPlane

```
public GameObject m_indicatorPlane
```

Field Value

GameObject

m_levelNormalizer

```
private double m_levelNormalizer
```

Field Value

[double](#) ↗

m_objs

```
private Material[] m_objs
```

Field Value

Material[]

m_start_time

```
private DateTime m_start_time
```

Field Value

[DateTime](#)

m_width

```
private readonly float m_width
```

Field Value

[float](#)

Methods

Start()

```
private UniTask Start()
```

Returns

UniTask

Update()

```
private void Update()
```

Namespace SamuraiSoccer.StageContents. BattlerDome

Classes

[BattleDomeManager](#)

Class BattleDomeManager

Namespace: [SamuraiSoccer.StageContents.BattlerDome](#)

Assembly: BattleDome.dll

```
public class BattleDomeManager : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← BattleDomeManager

Fields

audioSource

```
public AudioSource audioSource
```

Field Value

clearNum

```
private int clearNum
```

Field Value

[int](#)

conversationManager

```
public ConversationManager conversationManager
```

Field Value

[ConversationManager](#)

goalSound

`public AudioClip goalSound`

Field Value

AudioClip

resultSceneName

`public string resultSceneName`

Field Value

[string](#)

scenarioNum

`private int scenarioNum`

Field Value

[int](#)

startSound

`public AudioClip startSound`

Field Value

AudioClip

Methods

Clear()

```
private void Clear()
```

Conversation(int)

```
private UniTask Conversation(int num)
```

Parameters

num [int](#)

Returns

UniTask

GoalAction(GoalEventType)

```
private UniTask GoalAction(GoalEventType t)
```

Parameters

t [GoalEventType](#)

Returns

UniTask

SlowToWin()

勝った時にシーン遷移までゆっくりにして遷移させる

```
private UniTask SlowToWin()
```

Returns

UniTask

Start()

```
private void Start()
```

Namespace SamuraiSoccer.StageContents.China

Classes

[JiangshiPostProcessing](#)

キヨンシーシーンにおいてカメラにポストエフェクトをかける

[JiangshiSceneManager](#)

[KunfuReferee](#)

カンフーシーンのレフェリー

[KunfuSceneManager](#)

審判をカンフーに置き換える

[ModelDebug](#)

[Panda](#)

[PandaMaker](#)

[SoundBox](#)

音を鳴らして消えるオブジェクト

[SoundBoxUtil](#)

Enums

[Panda.State](#)

Class JiangshiPostProcessing

Namespace: [SamuraiSoccer.StageContents.China](#)

Assembly: SamuraiSoccer.StageContents.China.dll

キヨンシーシーンにおいてカメラにポストエフェクトをかける

```
public class JiangshiPostProcessing : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← JiangshiPostProcessing

Fields

m_material

```
[SerializeField]  
private Material m_material
```

Field Value

Material

Methods

OnRenderImage(RenderTexture, RenderTexture)

```
private void OnRenderImage(RenderTexture src, RenderTexture dest)
```

Parameters

src RenderTexture

dest RenderTexture

Start()

```
private void Start()
```

Class JiangshiSceneManager

Namespace: [SamuraiSoccer.StageContents.China](#)

Assembly: SamuraiSoccer.StageContents.China.dll

```
public class JiangshiSceneManager : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← JiangshiSceneManager

Fields

m_areaAngle

```
private int m_areaAngle
```

Field Value

[int](#)

m_areaSize

```
private int m_areaSize
```

Field Value

[int](#)

m_jianshi

```
private GameObject m_jianshi
```

Field Value

GameObject

m_prefab

```
[SerializeField]  
private GameObject m_prefab
```

Field Value

GameObject

m_referee

```
[SerializeField]  
private GameObject m_referee
```

Field Value

GameObject

m_refereeArea

```
[SerializeField]  
private RefereeArea m_refereeArea
```

Field Value

[RefereeArea](#)

m_refereeMove

```
[SerializeField]  
private RefereeMove m_refereeMove
```

Field Value

[RefereeMove](#)

Methods

Awake()

```
private void Awake()
```

Update()

```
private void Update()
```

Class KunfuReferee

Namespace: [SamuraiSoccer.StageContents.China](#)

Assembly: SamuraiSoccer.StageContents.China.dll

カンフーシーンのレフェリー

```
public class KunfuReferee : RefereeMove
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← [RefereeMove](#) ← KunfuReferee

Inherited Members

[RefereeMove.ball](#) , [RefereeMove.lookatspeed](#) , [RefereeMove.runningspeed](#) , [RefereeMove.radius](#) ,
[RefereeMove.rig](#) , [RefereeMove.refereeArea](#) , [RefereeMove.state](#) , [RefereeMove.anicon](#) ,
[RefereeMove.Update\(\)](#) , [RefereeMove.MoveAroundBall\(\)](#).

Methods

LookAtBall()

```
protected override void LookAtBall()
```

Start()

```
protected override void Start()
```

Class KunfuSceneManager

Namespace: [SamuraiSoccer.StageContents.China](#)

Assembly: SamuraiSoccer.StageContents.China.dll

審判をカンフーに置き換える

```
public class KunfuSceneManager : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← KunfuSceneManager

Fields

m_kunfu

```
private GameObject m_kunfu
```

Field Value

GameObject

m_prefab

```
[SerializeField]  
private GameObject m_prefab
```

Field Value

GameObject

m_referee

```
[SerializeField]  
private GameObject m_referee
```

Field Value

GameObject

Methods

Start()

```
private void Start()
```

Update()

```
private void Update()
```

Class ModelDebug

Namespace: [SamuraiSoccer.StageContents.China](#)

Assembly: SamuraiSoccer.StageContents.China.dll

```
public class ModelDebug : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← ModelDebug

Fields

prefabNameAndjsonName

```
public string prefabNameAndjsonName
```

Field Value

[string](#)

Methods

Awake()

```
private void Awake()
```

Class Panda

Namespace: [SamuraiSoccer.StageContents.China](#)

Assembly: SamuraiSoccer.StageContents.China.dll

```
public class Panda : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← Panda

Fields

m_anim

```
private Animator m_anim
```

Field Value

Animator

m_blood

```
[SerializeField]  
private GameObject m_blood
```

Field Value

GameObject

m_expandAmount

```
private float m_expandAmount
```

Field Value

[float](#) ↗

m_gameOverPanel

```
[SerializeField]  
private GameObject m_gameOverPanel
```

Field Value

GameObject

m_hit

```
private bool m_hit
```

Field Value

[bool](#) ↗

m_hitSound

```
[SerializeField]  
private AudioClip m_hitSound
```

Field Value

AudioClip

m_player

```
private GameObject m_player
```

Field Value

GameObject

m_resultSceneName

```
[SerializeField]  
private string m_resultSceneName
```

Field Value

[string](#)

m_speed

```
[SerializeField]  
private float m_speed
```

Field Value

[float](#)

m_state

```
private IReadOnlyReactiveProperty<Panda.State> m_state
```

Field Value

[IReadOnlyReactiveProperty<Panda.State>](#)

Methods

[GameOver\(\)](#)

```
private UniTask GameOver()
```

Returns

UniTask

OnTriggerEnter(Collider)

```
private void OnTriggerEnter(Collider other)
```

Parameters

other Collider

Start()

```
private void Start()
```

Update()

```
private void Update()
```

Enum Panda.State

Namespace: [SamuraiSoccer.StageContents.China](#)

Assembly: SamuraiSoccer.StageContents.China.dll

```
private enum Panda.State
```

Fields

Active = 0

Non_Interference = 1

Stop = 2

Vanish = 3

Class PandaMaker

Namespace: [SamuraiSoccer.StageContents.China](#)

Assembly: SamuraiSoccer.StageContents.China.dll

```
public class PandaMaker : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← PandaMaker

Fields

mIsActive

```
private IReadOnlyReactiveProperty<bool> mIsActive
```

Field Value

IReadOnlyReactiveProperty<[bool](#)>

m_maxSize

```
[SerializeField]  
private float m_maxSize
```

Field Value

[float](#)

m_minSize

```
[SerializeField]  
private float m_minSize
```

Field Value

[float](#)

m_panda

```
[SerializeField]  
private GameObject m_panda
```

Field Value

GameObject

Methods

PandaSpawn(CancellationToken)

```
private UniTask PandaSpawn(CancellationToken token)
```

Parameters

token [CancellationToken](#)

Returns

UniTask

Start()

```
private void Start()
```

Class SoundBox

Namespace: [SamuraiSoccer.StageContents.China](#)

Assembly: SamuraiSoccer.StageContents.China.dll

音を鳴らして消えるオブジェクト

```
public class SoundBox : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← SoundBox

Methods

StartSoundBox(AudioClip)

```
public void StartSoundBox(AudioClip clip)
```

Parameters

`clip` AudioClip

Vanish()

```
private void Vanish()
```

Class SoundBoxUtil

Namespace: [SamuraiSoccer.StageContents.China](#)

Assembly: SamuraiSoccer.StageContents.China.dll

```
public class SoundBoxUtil : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← SoundBoxUtil

Methods

SetSoundBox(Vector3, AudioClip)

```
public static void SetSoundBox(Vector3 position, AudioClip clip)
```

Parameters

position Vector3

clip AudioClip

Namespace SamuraiSoccer.StageContents.Conversation

Classes

[ConversationCharacter](#)

[ConversationCharacters](#)

[ConversationManager](#)

[ConversationText](#)

[SelectedConversationPublisher](#)

[StageConversationData](#)

[StageConversationDatas](#)

Enums

[CharacterName](#)

[EmotionType](#)

Enum CharacterName

Namespace: [SamuraiSoccer.StageContents.Conversation](#)

Assembly: Conversation.dll

```
[Serializable]
public enum CharacterName
```

Fields

AI = 5

xxx = 6

コクオー = 1

シドーシャ = 4

ショーグン = 0

ソーショキ = 2

ダイトリョー = 3

Class ConversationCharacter

Namespace: [SamuraiSoccer.StageContents.Conversation](#)

Assembly: Conversation.dll

```
[Serializable]
public class ConversationCharacter
```

Inheritance

[object](#) ← ConversationCharacter

Fields

m_characterName

```
public CharacterName m_characterName
```

Field Value

[CharacterName](#)

m_imageAngry

```
public Sprite m_imageAngry
```

Field Value

Sprite

m_imageFunny

```
public Sprite m_imageFunny
```

Field Value

Sprite

m_imageNormal

```
public Sprite m_imageNormal
```

Field Value

Sprite

m_imageSad

```
public Sprite m_imageSad
```

Field Value

Sprite

m_imageSilhouette

```
public Sprite m_imageSilhouette
```

Field Value

Sprite

Class ConversationCharacters

Namespace: [SamuraiSoccer.StageContents.Conversation](#)

Assembly: Conversation.dll

```
[CreateAssetMenu]  
public class ConversationCharacters : ScriptableObject
```

Inheritance

[object](#) ← Object ← ScriptableObject ← ConversationCharacters

Fields

m_conversationCharacters

```
public List<ConversationCharacter> m_conversationCharacters
```

Field Value

[List](#) <[ConversationCharacter](#)>

Class ConversationManager

Namespace: [SamuraiSoccer.StageContents.Conversation](#)

Assembly: Conversation.dll

```
public class ConversationManager : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← ConversationManager

Fields

m_brushPen

```
[SerializeField]  
private GameObject m_brushPen
```

Field Value

GameObject

m_characterImages

```
[SerializeField]  
[Tooltip("会話キャラクターの画像(左:0, 右:1)")]  
private Image[] m_characterImages
```

Field Value

Image[]

m_characterNameTexts

```
[SerializeField]
[Tooltip("会話キャラクターの名前(左:0, 右:1)")]
private Text[] m_characterNameTexts
```

Field Value

Text[]

m_conversationCharacters

```
[SerializeField]
private ConversationCharacters m_conversationCharacters
```

Field Value

[ConversationCharacters](#)

m_conversationContents

```
[SerializeField]
[Tooltip("会話文を表示するコンテンツ全体")]
private GameObject m_conversationContents
```

Field Value

GameObject

m_conversationDatas

```
[SerializeField]
private StageConversationDatas m_conversationDatas
```

Field Value

[StageConversationDatas](#)

m_conversationText

```
[SerializeField]  
private Text m_conversationText
```

Field Value

Text

m_initPos

```
private Vector3 m_initPos
```

Field Value

Vector3

m_isTouched

```
private bool m_isTouched
```

Field Value

[bool](#)

m_optionYesButtons

```
[SerializeField]  
private List<Button> m_optionYesButtons
```

Field Value

[List](#) <Button>

m_provider

```
[SerializeField]
private TouchProvider m_provider
```

Field Value

[TouchProvider](#)

m_scrollObject

```
[SerializeField]
private GameObject m_scrollObject
```

Field Value

GameObject

m_scrollScript

```
[SerializeField]
private ScrollScript m_scrollScript
```

Field Value

[ScrollScript](#)

m_textScroller

```
[SerializeField]
private TextScroller m_textScroller
```

Field Value

[TextScroller](#)

m_uiFade

```
[SerializeField]  
private UIFade m_uiFade
```

Field Value

[UIFade](#)

Methods

ActiveTextUI(bool)

会話プレハブのテキスト部分のSetActiveを一括で実施する

```
private void ActiveTextUI(bool setActive)
```

Parameters

setActive [bool](#) ↗

ChangeCharacterEmotion(Image, CharacterName, EmotionType)

会話キャラクターの画像を適切な感情のものに設定する

```
private void ChangeCharacterEmotion(Image image, CharacterName name,  
EmotionType emotionType)
```

Parameters

image [Image](#)

name [CharacterName](#)

emotionType [EmotionType](#)

ConversationProcess(int)

会話分の表示と再生

```
private UniTask ConversationProcess(int conversatioNum)
```

Parameters

conversatioNum [int](#)

再生する会話番号

Returns

UniTask

HopImage(Image, Vector3, CancellationToken)

```
private UniTask HopImage(Image image, Vector3 initPos, CancellationToken cancellationToken  
= default)
```

Parameters

image Image

initPos Vector3

cancellationToken [CancellationToken](#)

Returns

UniTask

PlayConversation(int)

会話コンテンツの起動

```
public UniTask PlayConversation(int conversationNum)
```

Parameters

conversationNum [int](#)

会話番号

Returns

UniTask

SetCharacterInfo(int)

会話キャラクター情報の追加

```
private void SetCharacterInfo(int conversationNum)
```

Parameters

conversationNum [int](#)

会話番号

Start()

```
public void Start()
```

Class ConversationText

Namespace: [SamuraiSoccer.StageContents.Conversation](#)

Assembly: Conversation.dll

```
[Serializable]
public class ConversationText
```

Inheritance

[object](#) ← ConversationText

Fields

is_option

```
public bool is_option
```

Field Value

[bool](#)

m_characterName

```
[Tooltip("話す人の名前")]
public CharacterName m_characterName
```

Field Value

[CharacterName](#)

m_motionType

```
[Tooltip("話し手の感情")]
public EmotionType m_motionType
```

Field Value

[EmotionType](#)

m_text

```
[Tooltip("話す内容")]
[TextArea(1, 5)]
public string m_text
```

Field Value

[string](#) ↗

Enum EmotionType

Namespace: [SamuraiSoccer.StageContents.Conversation](#)

Assembly: Conversation.dll

```
[Serializable]
public enum EmotionType
```

Fields

Angry = 3

Funny = 2

Normal = 0

Sad = 4

Silhouette = 1

Class SelectedConversationPublisher

Namespace: [SamuraiSoccer.StageContents.Conversation](#)

Assembly: SamuraiSoccer.StageContents.Conversation.dll

```
public class SelectedConversationPublisher : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← SelectedConversationPublisher

Fields

m_conversationManager

```
[SerializeField]  
private ConversationManager m_conversationManager
```

Field Value

[ConversationManager](#)

m_conversationNum

```
[SerializeField]  
private int m_conversationNum
```

Field Value

[int](#)

m_finishedConversation

```
private bool m_finishedConversation
```

Field Value

bool ↗

Methods

Onclick()

```
public void Onclick()
```

Class StageConversationData

Namespace: [SamuraiSoccer.StageContents.Conversation](#)

Assembly: Conversation.dll

```
[Serializable]
public class StageConversationData
```

Inheritance

[object](#) ← StageConversationData

Fields

m_conversationTexts

```
public List<ConversationText> m_conversationTexts
```

Field Value

[List](#) <[ConversationText](#)>

m_leftCharacterName

```
public CharacterName m_leftCharacterName
```

Field Value

[CharacterName](#)

m_rightCharacterName

```
public CharacterName m_rightCharacterName
```

Field Value

CharacterName

Class StageConversationDatas

Namespace: [SamuraiSoccer.StageContents.Conversation](#)

Assembly: Conversation.dll

```
[CreateAssetMenu]
public class StageConversationDatas : ScriptableObject
```

Inheritance

[Object](#) ← Object ← ScriptableObject ← StageConversationDatas

Fields

ConversationDatas

```
public List<StageConversationData> ConversationDatas
```

Field Value

[List](#) <[StageConversationData](#)>

Namespace SamuraiSoccer.StageContents.Result

Classes

[LoadAnyScene](#)

[LoadRetryScene](#)

[ResultBGM](#)

[ResultManager](#)

ゲームの結果を取得・表示

[SamuraiWordBase](#)

[StageDataSave](#)

Class LoadAnyScene

Namespace: [SamuraiSoccer.StageContents.Result](#)

Assembly: SamuraiSoccer.StageContents.Result.dll

```
public class LoadAnyScene : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← LoadAnyScene

Methods

LoadScene(string)

Scene遷移

```
public void LoadScene(string targetSceneName)
```

Parameters

targetSceneName [string](#)

ロード先のScene

LoadStartScene(string)

```
public void LoadStartScene(string targetSceneName)
```

Parameters

targetSceneName [string](#)

Class LoadRetryScene

Namespace: [SamuraiSoccer.StageContents.Result](#)

Assembly: SamuraiSoccer.StageContents.Result.dll

```
[RequireComponent(typeof(ResultManager))]  
public class LoadRetryScene : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← LoadRetryScene

Fields

m_stagePreviewDatas

```
[SerializeField]  
private StagePreviewDatas m_stagePreviewDatas
```

Field Value

[StagePreviewDatas](#)

Methods

LoadScene()

今遊んだSceneをStagePreviewDatasから取得しそのSceneへ移動

```
public void LoadScene()
```

Class ResultBGM

Namespace: [SamuraiSoccer.StageContents.Result](#)

Assembly: SamuraiSoccer.StageContents.Result.dll

```
public class ResultBGM : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← ResultBGM

Fields

m_cancellationTokenSource

```
private CancellationTokenSource m_cancellationTokenSource
```

Field Value

[CancellationTokenSource](#)

m_loseBGNum

```
[SerializeField]  
private int m_loseBGNum
```

Field Value

[int](#)

m_loseSENum

```
[SerializeField]  
private int m_loseSENum
```

Field Value

[int↗](#)

m_rm

```
[SerializeField]
private ResultManager m_rm
```

Field Value

[ResultManager](#)

m_winBGMNum

```
[SerializeField]
private int m_winBGMNum
```

Field Value

[int↗](#)

m_winSENum

```
[SerializeField]
private int m_winSENum
```

Field Value

[int↗](#)

Methods

PlayLoseBGM(CancellationToken)

```
private UniTask PlayLoseBGM(CancellationToken cancellationToken)
```

Parameters

cancellationToken [CancellationToken ↗](#)

Returns

UniTask

PlayWinBGM(CancellationToken)

```
private UniTask PlayWinBGM(CancellationToken cancellationToken)
```

Parameters

cancellationToken [CancellationToken ↗](#)

Returns

UniTask

SceneLoaded(Scene, LoadSceneMode)

```
private void SceneLoaded(Scene nextScene, LoadSceneMode mode)
```

Parameters

nextScene Scene

mode LoadSceneMode

Start()

```
private void Start()
```


Class ResultManager

Namespace: [SamuraiSoccer.StageContents.Result](#)

Assembly: SamuraiSoccer.StageContents.Result.dll

ゲームの結果を取得・表示

```
public class ResultManager : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← ResultManager

Fields

conversationManager

```
[SerializeField]  
private ConversationManager conversationManager
```

Field Value

[ConversationManager](#)

lastButtonTransform

```
[SerializeField]  
private RectTransform lastButtonTransform
```

Field Value

RectTransform

lastStageNum

```
[SerializeField]  
private int lastStageNum
```

Field Value

[int](#)

mainCamera

```
[SerializeField]  
private Camera mainCamera
```

Field Value

Camera

nextButtonTransform

```
[SerializeField]  
private RectTransform nextButtonTransform
```

Field Value

RectTransform

result

```
[SerializeField]  
private Text result
```

Field Value

Text

retryButtonTransform

```
[SerializeField]  
private RectTransform retryButtonTransform
```

Field Value

RectTransform

samuraiPhrase

```
[SerializeField]  
private Text samuraiPhrase
```

Field Value

Text

samuraiWordBase

```
[SerializeField]  
private SamuraiWordBase samuraiWordBase
```

Field Value

[SamuraiWordBase](#)

stageDataSave

```
[SerializeField]  
private StageDataSave stageDataSave
```

Field Value

[StageDataSave](#)

texts

```
[SerializeField]  
private List<Text> texts
```

Field Value

[List](#) <Text>

Properties

ResultState

```
public GameResult ResultState { get; private set; }
```

Property Value

[GameResult](#)

Methods

Lose()

負けた時は黒地に白文字になる

```
private void Lose()
```

Start()

```
private void Start()
```

Win()

勝った時は白地に黒文字になってセーブ処理が発生する

```
private void Win()
```

Class SamuraiWordBase

Namespace: [SamuraiSoccer.StageContents.Result](#)

Assembly: SamuraiSoccer.StageContents.Result.dll

```
[CreateAssetMenu(menuName = "ScrptableObject/CreateSamuraiWordBase", fileName  
= "SamuraiWordBase")]  
public class SamuraiWordBase : ScriptableObject
```

Inheritance

[object](#) ← Object ← ScriptableObject ← SamuraiWordBase

Fields

samuraiwords

```
[TextArea(1, 3)]  
public List<string> samuraiwords
```

Field Value

[List](#) <[string](#)>

Class StageDataSave

Namespace: [SamuraiSoccer.StageContents.Result](#)

Assembly: SamuraiSoccer.StageContents.Result.dll

```
public class StageDataSave : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← StageDataSave

Methods

ResetData()

セーブデータを初期化する(ResetだとMonoBehaviourと名前が被るから命名変更)

```
public void ResetData()
```

Save(int)

クリア番号が過去のセーブデータ以上だったらセーブデータを更新する

```
public bool Save(int clearNumber)
```

Parameters

clearNumber [int](#)

クリア番号

Returns

[bool](#)

セーブしたかどうか

Namespace SamuraiSoccer.StageContents. Russian Classes

[IceStageManager](#)

ロシアステージのステージ設定を行う。

[Snow](#)

雪を制御する。

Class IceStageManager

Namespace: [SamuraiSoccer.StageContents.Russian](#)

Assembly: SamuraiSoccer.StageContents.Russian.dll

ロシアステージのステージ設定を行う。

```
[DefaultExecutionOrder(-1)]  
public class IceStageManager : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← IceStageManager

Fields

eclanoplan

```
public GameObject eclanoplan
```

Field Value

GameObject

greenMaterial

```
public Material greenMaterial
```

Field Value

Material

greenTexture

```
public Texture greenTexture
```

Field Value

Texture

iceMaterial

```
public Material iceMaterial
```

Field Value

Material

iceObject

```
public GameObject iceObject
```

Field Value

GameObject

iceTexture

```
public Texture iceTexture
```

Field Value

Texture

mountObject

```
public GameObject mountObject
```

Field Value

GameObject

no

```
public static int no
```

Field Value

[int ↗](#)

stageManager

```
public StagePrefabManager stageManager
```

Field Value

[StagePrefabManager](#)

Methods

Start()

```
private void Start()
```

Class Snow

Namespace: [SamuraiSoccer.StageContents.Russian](#)

Assembly: SamuraiSoccer.StageContents.Russian.dll

雪を制御する。

```
public class Snow : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← Snow

Fields

audioSource

```
public AudioSource audioSource
```

Field Value

beforePoint

```
private Vector3 beforePoint
```

Field Value

Vector3

damage

```
[SerializeField]  
private float damage
```

Field Value

[float](#)

gameover

```
public Image gameover
```

Field Value

Image

isPlaying

```
private bool isPlaying
```

Field Value

[bool](#)

mainCamera

```
public Transform mainCamera
```

Field Value

Transform

particle

```
public ParticleSystem particle
```

Field Value

ParticleSystem

samurai

```
public Transform samurai
```

Field Value

Transform

snow

```
public Transform snow
```

Field Value

Transform

time

```
private float time
```

Field Value

[float](#)

Methods

GameOver()

凍ってゲームオーバー時の処理。

```
private UniTask GameOver()
```

Returns

UniTask

OnPauseSnow(bool)

```
private void OnPauseSnow(bool isPause)
```

Parameters

isPause bool

OnPlaySnow(Unit)

```
private void OnPlaySnow(Unit unit)
```

Parameters

unit Unit

OnReset(Unit)

```
private void OnReset(Unit unit)
```

Parameters

unit Unit

OnUpdateDuringPlay(long)

```
private void OnUpdateDuringPlay(long __)
```

Parameters

[long ↗](#)

Start()

```
private void Start()
```

TimerReset()

```
public void TimerReset()
```

Update()

```
private void Update()
```

Namespace SamuraiSoccer.StageContents. StageSelect

Classes

[Bezier2Order3Points](#)

制御点三点の二次ベジエ曲線作製クラス。

[LastBattleTest](#)

[ScrollIcon](#)

[ScrollManager](#)

[ScrollMove](#)

[StageCamera](#)

各ステージをフォーカスするカメラ

[StageIcon](#)

[StagePreviewData](#)

[StagePreviewDatas](#)

[StagePreviewMove](#)

[StageScroll](#)

[StageScrollScroller](#)

[StageSelectBGM](#)

ステージセレクトのBGMを監理。

[StageSelectCameraMove](#)

[StageStateInit](#)

[WorldMapMove](#)

ワールドマップを移動するクラス。 関数を呼んで移動させる。

Enums

[StageState](#)

ステージの状態 NotPlayable:プレイ不可 Playable:プレイ可能 Cleared:クリア済み

Class Bezier2Order3Points

Namespace: [SamuraiSoccer.StageContents.StageSelect](#)

Assembly: SamuraiSoccer.StageContents.StageSelect.dll

制御点三点の二次ベジエ曲線作製クラス。

```
public sealed class Bezier2Order3Points
```

Inheritance

[object](#) ← Bezier2Order3Points

Constructors

Bezier2Order3Points(Vector2, Vector2, Vector2)

ベジエ曲線の作製。

```
public Bezier2Order3Points(Vector2 start, Vector2 medium, Vector2 end)
```

Parameters

start Vector2

始点

medium Vector2

制御点

end Vector2

終点

Fields

m_InCombi

```
private List<float> m_lnCombi
```

Field Value

[List](#) <[float](#)>

m_points

```
private List<Vector2> m_points
```

Field Value

[List](#) <[Vector2](#)>

Methods

GetPoint(float)

ベジエ曲線の点を取得。

```
public Vector2 GetPoint(float t)
```

Parameters

t [float](#)

地点

Returns

[Vector2](#)

点

Exceptions

[ArgumentException](#)

範囲外のtを入れたとき投げられる。

Class LastBattleTest

Namespace: [SamuraiSoccer.StageContents.StageSelect](#)

Assembly: SamuraiSoccer.StageContents.StageSelect.dll

```
public class LastBattleTest : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← LastBattleTest

Fields

stageIcon

```
[SerializeField]  
private StageIcon stageIcon
```

Field Value

[StageIcon](#)

Methods

Start()

```
private void Start()
```

Update()

```
private void Update()
```

Class ScrollIcon

Namespace: [SamuraiSoccer.StageContents.StageSelect](#)

Assembly: SamuraiSoccer.StageContents.StageSelect.dll

```
public class ScrollIcon : StageIcon
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← [StageIcon](#) ← ScrollIcon

Inherited Members

[StageIcon.m_stageNumber](#), [StageIcon.m_slashSE](#), [StageIcon.StageNumber](#), [StageIcon.State](#),
[StageIcon.OnClick\(\)](#).

Fields

m_colorCoef

```
[SerializeField]  
[Range(0, 1)]  
private float m_colorCoef
```

Field Value

[float](#)

Methods

Start()

```
private void Start()
```

Class ScrollManager

Namespace: [SamuraiSoccer.StageContents.StageSelect](#)

Assembly: SamuraiSoccer.StageContents.StageSelect.dll

```
public class ScrollManager : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← ScrollManager

Fields

m_currentStage

```
private Stage m_currentStage
```

Field Value

[Stage](#)

m_scrollMoves

```
public List<ScrollMove> m_scrollMoves
```

Field Value

[List](#) <[ScrollMove](#)>

m_stageSelectBGM

```
[SerializeField]  
private StageSelectBGM m_stageSelectBGM
```

Field Value

[StageSelectBGM](#)

m_worldMapMove

```
[SerializeField]  
private WorldMapMove m_worldMapMove
```

Field Value

[WorldMapMove](#)

Methods

Start()

```
private void Start()
```

Class ScrollMove

Namespace: [SamuraiSoccer.StageContents.StageSelect](#)

Assembly: SamuraiSoccer.StageContents.StageSelect.dll

```
public class ScrollMove : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← ScrollMove

Fields

m_cts

```
private CancellationTokenSource m_cts
```

Field Value

[CancellationTokenSource](#)

m_goalX

```
[SerializeField]  
private float m_goalX
```

Field Value

[float](#)

m_hidePanel

```
[SerializeField]  
private HidePanel m_hidePanel
```

Field Value

[HidePanel](#)

m_initPos

```
private Vector3 m_initPos
```

Field Value

Vector3

m_scrollObject

```
[SerializeField]  
private GameObject m_scrollObject
```

Field Value

GameObject

m_scrollObjects

```
[SerializeField]  
private GameObject m_scrollObjects
```

Field Value

GameObject

m_scrollScript

```
[SerializeField]  
private ScrollScript m_scrollScript
```

Field Value

[ScrollScript](#)

m_stage

```
[SerializeField]  
private Stage m_stage
```

Field Value

[Stage](#)

Properties

Stage

```
public Stage Stage { get; }
```

Property Value

[Stage](#)

Methods

Move()

```
public UniTask Move()
```

Returns

UniTask

ResetMove()

```
public void ResetMove()
```

Start()

```
private void Start()
```

Class StageCamera

Namespace: [SamuraiSoccer.StageContents.StageSelect](#)

Assembly: SamuraiSoccer.StageContents.StageSelect.dll

各ステージをフォーカスするカメラ

```
[Serializable]
public class StageCamera
```

Inheritance

[Object](#) ← StageCamera

Fields

camera

```
public CinemachineVirtualCamera camera
```

Field Value

CinemachineVirtualCamera

stage

```
public Stage stage
```

Field Value

[Stage](#)

Class StageIcon

Namespace: [SamuraiSoccer.StageContents.StageSelect](#)

Assembly: SamuraiSoccer.StageContents.StageSelect.dll

```
public class StageIcon : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← StageIcon

Derived

[ScrollIcon](#)

Fields

m_slashSE

```
private int m_slashSE
```

Field Value

[int](#)

m_stageNumber

```
[SerializeField]  
private int m_stageNumber
```

Field Value

[int](#)

Properties

StageNumber

Inspectorから設定するこのステージの番号

```
public int StageNumber { get; }
```

Property Value

[int ↗](#)

State

```
public StageState State { get; set; }
```

Property Value

[StageState](#)

Methods

OnClick()

```
public void OnClick()
```

Start()

```
private void Start()
```

Class StagePreviewData

Namespace: [SamuraiSoccer.StageContents.StageSelect](#)

Assembly: SamuraiSoccer.StageContents.StageSelect.dll

```
[Serializable]
public class StagePreviewData
```

Inheritance

[object](#) ← StagePreviewData

Fields

fieldNumber

```
public int fieldNumber
```

Field Value

[int](#)

gameScene

```
public SceneObject gameScene
```

Field Value

[SceneObject](#)

groundColor

```
public int groundNumber
```

Field Value

[int](#) ↗

name

`public string name`

Field Value

[string](#) ↗

opponentType

`public string opponentType`

Field Value

[string](#) ↗

previewName

`public string previewName`

Field Value

[string](#) ↗

stageImage

`public Sprite stageImage`

Field Value

stageNumber

```
public int stageNumber
```

Field Value

[int ↗](#)

summary

```
[TextArea(1, 5)]  
public string summary
```

Field Value

[string ↗](#)

Class StagePreviewDatas

Namespace: [SamuraiSoccer.StageContents.StageSelect](#)

Assembly: SamuraiSoccer.StageContents.StageSelect.dll

```
[CreateAssetMenu]
public class StagePreviewDatas : ScriptableObject
```

Inheritance

[Object](#) ← Object ← ScriptableObject ← StagePreviewDatas

Fields

stageSelectList

```
public List<StagePreviewData> stageSelectList
```

Field Value

[List](#) <[StagePreviewData](#)>

Class StagePreviewMove

Namespace: [SamuraiSoccer.StageContents.StageSelect](#)

Assembly: SamuraiSoccer.StageContents.StageSelect.dll

```
public class StagePreviewMove : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← StagePreviewMove

Fields

m_currentStagePreviewData

```
private StagePreviewData m_currentStagePreviewData
```

Field Value

[StagePreviewData](#)

m_previewObject

```
[SerializeField]  
private GameObject m_previewObject
```

Field Value

GameObject

m_slashSE

```
private int m_slashSE
```

Field Value

[int](#)

m_stagelImage

```
[SerializeField]  
private Image m_stageImage
```

Field Value

Image

m_stageName

```
[SerializeField]  
private Text m_stageName
```

Field Value

Text

m_stagePreviewDatas

```
[SerializeField]  
private StagePreviewDatas m_stagePreviewDatas
```

Field Value

[StagePreviewDatas](#)

m_stageSummary

```
[SerializeField]
```

```
private Text m_stageSummary
```

Field Value

Text

Methods

Close()

戻るボタンでPreview表示を閉じる

```
public void Close()
```

Start()

```
private void Start()
```

StartGame()

開戦ボタンで試合Sceneへ遷移

```
public void StartGame()
```

Class StageScroll

Namespace: [SamuraiSoccer.StageContents.StageSelect](#)

Assembly: SamuraiSoccer.StageContents.StageSelect.dll

```
public class StageScroll : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← StageScroll

Fields

m_conversationPublisher

```
[SerializeField]  
private SelectedConversationPublisher m_conversationPublisher
```

Field Value

[SelectedConversationPublisher](#)

m_image

```
[SerializeField]  
private Image m_image
```

Field Value

Image

m_imageRect

```
[SerializeField]  
private RectTransform m_imageRect
```

Field Value

RectTransform

m_publisher

```
[SerializeField]  
private SelectedStagePublisher m_publisher
```

Field Value

[SelectedStagePublisher](#)

m_scrollIcon

```
[SerializeField]  
private ScrollIcon m_scrollIcon
```

Field Value

[ScrollIcon](#)

m_stage

```
[SerializeField]  
private Stage m_stage
```

Field Value

[Stage](#)

m_viewPort

```
[SerializeField]
```

```
private RectTransform m_viewPort
```

Field Value

RectTransform

Properties

Stage

```
public Stage Stage { get; }
```

Property Value

[Stage](#)

Methods

OnClick()

```
public void OnClick()
```

Update()

```
private void Update()
```

Class StageScrollScroller

Namespace: [SamuraiSoccer.StageContents.StageSelect](#)

Assembly: SamuraiSoccer.StageContents.StageSelect.dll

```
public class StageScrollScroller : ScrollRect, IInitializePotentialDragHandler,  
IBeginDragHandler, IEndDragHandler, IDragHandler, IScrollHandler, IEventSystemHandler,  
ICanvasElement, ILayoutElement, ILayoutGroup, ILayoutController
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← UIBehaviour ← ScrollRect ← StageScrollScroller

Implements

IInitializePotentialDragHandler, IBeginDragHandler, IEndDragHandler, IDragHandler, IScrollHandler, IEventSystemHandler, ICanvasElement, ILayoutElement, ILayoutGroup, ILayoutController

Inherited Members

ScrollRect.m_ContentStartPosition , ScrollRect.m_ContentBounds , ScrollRect.Rebuild(CanvasUpdate) ,
ScrollRect.LayoutComplete() , ScrollRect.GraphicUpdateComplete() , ScrollRect.OnEnable() ,
ScrollRect.OnDisable() , ScrollRect.IsActive() , ScrollRect.StopMovement() ,
ScrollRect.OnScroll(PointerEventData) , ScrollRect.OnInitializePotentialDrag(PointerEventData) ,
ScrollRect.SetContentAnchoredPosition(Vector2) , ScrollRect.LateUpdate() , ScrollRect.UpdatePrevData() ,
[ScrollRect.SetNormalizedPosition\(float, int\)](#) , ScrollRect.OnRectTransformDimensionsChange() ,
ScrollRect.CalculateLayoutInputHorizontal() , ScrollRect.CalculateLayoutInputVertical() ,
ScrollRect.SetLayoutHorizontal() , ScrollRect.SetLayoutVertical() , ScrollRect.UpdateBounds() ,
ScrollRect.SetDirty() , ScrollRect.SetDirtyCaching() , ScrollRect.OnValidate() ,
ScrollRect.ICanvasElement.get_transform() , ScrollRect.content , ScrollRect.horizontal , ScrollRect.vertical ,
ScrollRect.movementType , ScrollRect.elasticity , ScrollRect.inertia , ScrollRect.decelerationRate ,
ScrollRect.scrollSensitivity , ScrollRect.viewport , ScrollRect.horizontalScrollbar ,
ScrollRect.verticalScrollbar , ScrollRect.horizontalScrollbarVisibility , ScrollRect.verticalScrollbarVisibility ,
ScrollRect.horizontalScrollbarSpacing , ScrollRect.verticalScrollbarSpacing , ScrollRect.onValueChanged ,
ScrollRect.viewRect , ScrollRect.velocity , ScrollRect.normalizedPosition ,
ScrollRect.horizontalNormalizedPosition , ScrollRect.verticalNormalizedPosition , ScrollRect.minWidth ,
ScrollRect.preferredWidth , ScrollRect.flexibleWidth , ScrollRect.minLength , ScrollRect.preferredHeight ,
ScrollRect.flexibleHeight , ScrollRect.layoutPriority , UIBehaviour.Awake() , UIBehaviour.OnDestroy() ,
UIBehaviour.Reset() , UIBehaviour.OnBeforeTransformParentChanged() ,
UIBehaviour.OnTransformParentChanged() , UIBehaviour.OnDidApplyAnimationProperties()

UIBehaviour.OnCanvasGroupChanged() , UIBehaviour.OnCanvasHierarchyChanged() ,
UIBehaviour.IsDestroyed()

Fields

m_cancellationTokenSource

```
private CancellationTokenSource m_cancellationTokenSource
```

Field Value

[CancellationTokenSource](#)

m_isDrugging

```
private bool m_isDrugging
```

Field Value

[bool](#)

m_pageNumber

```
private int m_pageNumber
```

Field Value

[int](#)

m_selectingStageReactiveProperty

```
private static ReactiveProperty<Stage> m_selectingStageReactiveProperty
```

Field Value

ReactiveProperty<[Stage](#)>

m_slideCount

```
private int m_slideCount
```

Field Value

[int](#) ↗

Properties

SelectedStage

選択中のStageが変更されたときに発行するReactiveProperty

```
public static IReadOnlyReactiveProperty<Stage> SelectedStage { get; }
```

Property Value

IReadOnlyReactiveProperty<[Stage](#)>

Methods

GetSelectingStage()

スライダーの中心にある国を取得

```
private Stage GetSelectingStage()
```

Returns

[Stage](#)

OnBeginDrag(PointerEventData)

```
public override void OnBeginDrag(PointerEventData data)
```

Parameters

data PointerEventData

OnDrag(PointerEventData)

```
public override void OnDrag(PointerEventData eventData)
```

Parameters

eventData PointerEventData

OnEndDrag(PointerEventData)

```
public override void OnEndDrag(PointerEventData data)
```

Parameters

data PointerEventData

ScrollRoop()

```
private void ScrollRoop()
```

SliderAdjust(CancellationToken)

```
private UniTask SliderAdjust(CancellationToken cancellationToken)
```

Parameters

cancellationToken [CancellationToken](#)

Returns

UniTask

Start()

```
protected override void Start()
```

Update()

```
private void Update()
```

setScrollPos()

```
private void setScrollPos()
```

Class StageSelectBGM

Namespace: [SamuraiSoccer.StageContents.StageSelect](#)

Assembly: SamuraiSoccer.StageContents.StageSelect.dll

ステージセレクトのBGMを監理。

```
public class StageSelectBGM : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← StageSelectBGM

Fields

cn

```
[SerializeField]  
private AudioSource cn
```

Field Value

current

```
private Stage current
```

Field Value

[Stage](#)

gb

```
[SerializeField]  
private AudioSource gb
```

Field Value

isNotCleared

```
private bool isNotCleared
```

Field Value

bool ↗

jp

```
[SerializeField]  
private AudioSource jp
```

Field Value

ru

```
[SerializeField]  
private AudioSource ru
```

Field Value

start

```
[SerializeField]
private AudioSource start
```

Field Value

us

```
[SerializeField]
private AudioSource us
```

Field Value

Methods

ChangeBGM(Stage, float)

```
public void ChangeBGM(Stage next, float delayTime)
```

Parameters

next [Stage](#)

delayTime [float](#)

FadeIn(AudioSource, float, CancellationToken)

```
private UniTask FadeIn(AudioSource source, float delayTime, CancellationToken token)
```

Parameters

source AudioSource

delayTime [float](#)

token [CancellationToken](#)

Returns

UniTask

FadeOut(AudioSource, float, CancellationToken)

```
private UniTask FadeOut(AudioSource source, float delayTime, CancellationToken token)
```

Parameters

source AudioSource

delayTime [float](#)

token [CancellationToken](#)

Returns

UniTask

Start()

```
private void Start()
```

Class StageSelectCameraMove

Namespace: [SamuraiSoccer.StageContents.StageSelect](#)

Assembly: SamuraiSoccer.StageContents.StageSelect.dll

```
public class StageSelectCameraMove : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← StageSelectCameraMove

Fields

m_canTouchWorld

```
private bool m_canTouchWorld
```

Field Value

[bool](#)

m_cinemachineBrain

```
[SerializeField]  
private CinemachineBrain m_cinemachineBrain
```

Field Value

CinemachineBrain

m_stageCameras

```
[SerializeField]  
private List<StageCamera> m_stageCameras
```

Field Value

[List](#) <[StageCamera](#)>

Methods

ChangeCameraView(Stage)

指定したステージのカメラ優先度を高くしてフォーカスする

```
public UniTask ChangeCameraView(Stage stage)
```

Parameters

stage [Stage](#)

指定するステージ

Returns

[UniTask](#)

Start()

```
private void Start()
```

Enum StageState

Namespace: [SamuraiSoccer.StageContents.StageSelect](#)

Assembly: SamuraiSoccer.StageContents.StageSelect.dll

ステージの状態 NotPlayable:プレイ不可 Playable:プレイ可能 Cleared:クリア済み

```
public enum StageState
```

Fields

Cleared = 2

NotPlayable = 0

Playable = 1

Class StageStateInit

Namespace: [SamuraiSoccer.StageContents.StageSelect](#)

Assembly: SamuraiSoccer.StageContents.StageSelect.dll

```
public class StageStateInit : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← StageStateInit

Fields

m_stagelcon

```
[SerializeField]  
private StageIcon[] m_stageIcon
```

Field Value

[StageIcon\[\]](#)

Methods

Awake()

```
private void Awake()
```

Class WorldMapMove

Namespace: [SamuraiSoccer.StageContents.StageSelect](#)

Assembly: SamuraiSoccer.StageContents.StageSelect.dll

ワールドマップを移動するクラス。 関数を呼んで移動させる。

```
public class WorldMapMove : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← WorldMapMove

Fields

accMax

```
[SerializeField]  
private float accMax
```

Field Value

[float](#)

displaySize

```
[SerializeField]  
private float displaySize
```

Field Value

[float](#)

m_currentPoint

```
private Vector2 m_currentPoint
```

Field Value

Vector2

m_currentSize

```
private float m_currentSize
```

Field Value

[float](#)

m_isFloatingStop

```
private bool m_isFloatingStop
```

Field Value

[bool](#)

m_moving

```
private Stack<Stage> m_moving
```

Field Value

[Stack](#) <[Stage](#)>

m_pointHistory

```
private Queue<Vector2> m_pointHistory
```

Field Value

[Queue](#)<Vector2>

m_selectState

```
private Stage m_selectState
```

Field Value

[Stage](#)

m_velocity

```
private Vector2 m_velocity
```

Field Value

Vector2

m_worldPoint

```
private static readonly IReadOnlyDictionary<Stage, Vector2> m_worldPoint
```

Field Value

[IReadOnlyDictionary](#)<[Stage](#), Vector2>

m_worldSize

```
private static readonly IReadOnlyDictionary<Stage, float> m_worldSize
```

Field Value

[IReadOnlyDictionary](#)<[Stage](#), [float](#)>

moveTime

```
[SerializeField]  
private float moveTime
```

Field Value

[float](#)

worldMap

```
[SerializeField]  
private RectTransform worldMap
```

Field Value

RectTransform

Methods

Awake()

```
private void Awake()
```

FixedUpdate()

```
private void FixedUpdate()
```

GetObjectPoint(Vector2, Vector2)

目的地を取得.

```
private Vector2 GetObjectPoint(Vector2 src, Vector2 dst)
```

Parameters

src Vector2

出発点.

dst Vector2

目的地.

Returns

Vector2

目的地.

GoTo(Stage)

指定した国へ移動.

```
public UniTask GoTo(Stage country)
```

Parameters

country Stage

指定した国

Returns

UniTask

SetPosition(Vector2, float)

マップを移動。

```
private void SetPosition(Vector2 localPos, float localSize)
```

Parameters

localPos Vector2

マップ上のピクセル位置。

localSize [float](#)

表示マップの1辺のピクセル。

Start()

```
private void Start()
```

ToFloating(CancellationToken)

フロー-ティング状態にする。

```
public UniTask ToFloating(CancellationToken token = default)
```

Parameters

token [CancellationToken](#)

キャンセル用トークン。

Returns

UniTask

Namespace SamuraiSoccer.StageContents.UK

Classes

[ShotObjectScript](#)

[StrikeObjectScript](#)

Class ShotObjectScript

Namespace: [SamuraiSoccer.StageContents.UK](#)

Assembly: SamuraiSoccer.StageContents.UK.dll

```
public class ShotObjectScript : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← ShotObjectScript

Fields

dangerAreaLength

```
[SerializeField]  
private float dangerAreaLength
```

Field Value

[float](#)

groundWidth

```
private float groundWidth
```

Field Value

[float](#)

isActive

```
private bool isActive
```

Field Value

[bool](#) ↗

isEnd

```
private bool isEnd
```

Field Value

[bool](#) ↗

movedLength

```
private float movedLength
```

Field Value

[float](#) ↗

soundEffect

```
[SerializeField]  
private AudioSource soundEffect
```

Field Value

velocity

```
[SerializeField]  
private float velocity
```

Field Value

[float](#)

velocity0

```
private float velocity0
```

Field Value

[float](#)

Properties

rotateVec

```
private Vector3 rotateVec { get; }
```

Property Value

Vector3

Methods

BlowAway(GameObject, CancellationToken)

```
private UniTask BlowAway(GameObject player, CancellationToken cancellationToken = default)
```

Parameters

player GameObject

cancellationToken [CancellationToken](#)

Returns

Move()

```
private void Move()
```

Start()

```
private void Start()
```

Class StrikeObjectScript

Namespace: [SamuraiSoccer.StageContents.UK](#)

Assembly: SamuraiSoccer.StageContents.UK.dll

```
public class StrikeObjectScript : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← StrikeObjectScript

Fields

Cannons

```
[SerializeField]  
private Transform[] Cannons
```

Field Value

Transform[]

ShotObject

```
[SerializeField]  
private GameObject ShotObject
```

Field Value

GameObject

elapsedTime

```
private float elapsedTime
```

Field Value

[float](#) ↗

index

```
private int index
```

Field Value

[int](#) ↗

maxIndex

```
private int maxIndex
```

Field Value

[int](#) ↗

player

```
[SerializeField]  
private Transform player
```

Field Value

Transform

shotInterval

```
[SerializeField]  
private float shotInterval
```

Field Value

[float](#) ↗

Methods

Shot()

```
private void Shot()
```

Start()

```
private void Start()
```

Namespace SamuraiSoccer.StageContents.USA

Classes

[BulletFunction](#)

弾丸との衝突の処理,自滅処理

[CallStatueOfLiberty](#)

[CollisionWithStatue](#)

自由の女神との衝突の処理

[FireHypnoticBullets](#)

敵選手に付けて弾を撃たせる

[StatueMove](#)

自由の女神が上がってきたり倒れる動きの監視+実効

Class BulletFunction

Namespace: [SamuraiSoccer.StageContents.USA](#)

Assembly: SamuraiSoccer.StageContents.USA.dll

弾丸との衝突の処理,自滅処理

```
public class BulletFunction : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← BulletFunction

Fields

m_damageSEIndex

```
[SerializeField]  
private int m_damageSEIndex
```

Field Value

[int](#)

m_onTriggerDisposable

```
private IDisposable m_onTriggerDisposable
```

Field Value

[IDisposable](#)

m_rb

```
[SerializeField]  
private Rigidbody m_rb
```

Field Value

Rigidbody

m_resultSceneName

```
[SerializeField]  
private string m_resultSceneName
```

Field Value

[string](#)

m_timer

```
private float m_timer
```

Field Value

[float](#)

m_tmpVelocity

```
private Vector3 m_tmpVelocity
```

Field Value

Vector3

Methods

DestroyTimer()

タイマーに数字を加え続けて一定時間たつとオブジェクト消去

```
private void DestroyTimer()
```

PauseMove(bool)

ポーズ時の処理

```
private void PauseMove(bool ispause)
```

Parameters

ispause bool ↗

true:一時停止, false:解除

Start()

```
private void Start()
```

StopFunction()

プレイヤーとの衝突処理の購読を終了することで機能を停止

```
private void StopFunction()
```

Class CallStatueOfLiberty

Namespace: [SamuraiSoccer.StageContents.USA](#)

Assembly: SamuraiSoccer.StageContents.USA.dll

```
public class CallStatueOfLiberty : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← CallStatueOfLiberty

Fields

StatuePrefab

```
[SerializeField]  
private GameObject StatuePrefab
```

Field Value

GameObject

Methods

Start()

```
private void Start()
```

Class CollisionWithStatue

Namespace: [SamuraiSoccer.StageContents.USA](#)

Assembly: SamuraiSoccer.StageContents.USA.dll

自由の女神との衝突の処理

```
public class CollisionWithStatue : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← CollisionWithStatue

Fields

m_damageSEIndex

```
[SerializeField]  
private int m_damageSEIndex
```

Field Value

[int](#)

m_isActive

```
private bool m_isActive
```

Field Value

[bool](#)

m_resultSceneName

```
[SerializeField]  
private string m_resultSceneName
```

Field Value

[string](#)

m_statueMove

```
[SerializeField]  
private StatueMove m_statueMove
```

Field Value

[StatueMove](#)

Methods

ResetBall(GameObject)

ボールが押しつぶされて貫通した場合にコート内に戻す

```
private UniTask ResetBall(GameObject Ball)
```

Parameters

Ball GameObject

Returns

UniTask

Start()

```
private void Start()
```

Class FireHypnoticBullets

Namespace: [SamuraiSoccer.StageContents.USA](#)

Assembly: SamuraiSoccer.StageContents.USA.dll

敵選手に付けて弾を撃たせる

```
public class FireHypnoticBullets : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← FireHypnoticBullets

Fields

HypnoticBullets

```
[SerializeField]  
private GameObject HypnoticBullets
```

Field Value

GameObject

m_fireDuration

```
[SerializeField]  
private float m_fireDuration
```

Field Value

[float](#)

m_samurai

```
[SerializeField]  
private GameObject m_samurai
```

Field Value

GameObject

Methods

FireBullet()

弾を撃つ、次の発射間隔をランダム指定

```
private void FireBullet()
```

Start()

```
private void Start()
```

Class StatueMove

Namespace: [SamuraiSoccer.StageContents.USA](#)

Assembly: SamuraiSoccer.StageContents.USA.dll

自由の女神が上がってきで倒れる動きの監視+実効

```
public class StatueMove : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← StatueMove

Fields

m_bodyObj

```
[SerializeField]  
private GameObject m_bodyObj
```

Field Value

GameObject

m_shadeObj

```
[SerializeField]  
private GameObject m_shadeObj
```

Field Value

GameObject

m_spriteRenderer

```
[SerializeField]  
private SpriteRenderer m_spriteRenderer
```

Field Value

SpriteRenderer

m_time

```
private float m_time
```

Field Value

[float](#)

Properties

CurrentStatueMode

自由の女神の状態を管理する内部変数

```
public StatueMode CurrentStatueMode { get; private set; }
```

Property Value

[StatueMode](#)

Methods

MoveStatue()

自由の女神が移動して倒れ、その場で制止する。併せて影が点滅する。終了後削除される。

```
private void MoveStatue()
```

Start()

```
private void Start()
```

Namespace SamuraiSoccer.UI

Classes

[AttackButton](#)

[BlackoutPanel](#)

[ExceptionLogUI](#)

[HidePanel](#)

巻物の右側をHideパネルで隠しておく UIのレンダー順はいじれなかったのでスクリプト側で解決した

[OpponentNumber](#)

[PauseButton](#)

[ScrollScript](#)

巻物をなめらかに移動させる

[SelectedStagePublisher](#)

[SlidePad](#)

[TargetWorldData](#)

[Test_ChargeAttackButton](#)

[TextScroller](#)

UIテキストの一文字ずつ送る動作を行う。

[TimerScript](#)

[TouchProvider](#)

[UIFade](#)

Class AttackButton

Namespace: [SamuraiSoccer.UI](#)

Assembly: SamuraiSoccer.UI.dll

```
public class AttackButton : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← AttackButton

Fields

CHARGETIME

```
private float CHARGETIME
```

Field Value

[float](#)

m_isCharging

```
private bool m_isCharging
```

Field Value

[bool](#)

m_pushTime

```
private float m_pushTime
```

Field Value

[float](#)

Methods

OnLeaveAttackButton()

```
public void OnLeaveAttackButton()
```

OnPushAttackButton()

```
public void OnPushAttackButton()
```

Update()

```
private void Update()
```

Class BlackoutPanel

Namespace: [SamuraiSoccer.UI](#)

Assembly: SamuraiSoccer.UI.dll

```
public class BlackoutPanel : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← BlackoutPanel

Fields

m_panellImage

```
[SerializeField]  
private Image m_panelImage
```

Field Value

Image

Methods

Blackout(float)

試合画面の暗転を行う

```
public UniTask Blackout(float totaltimesec)
```

Parameters

totaltimesec [float](#)

暗転の全体時間

Returns

UniTask

Start()

```
private void Start()
```

Class ExeptionLogUI

Namespace: [SamuraiSoccer.UI](#)

Assembly: SamuraiSoccer.UI.dll

```
public class ExeptionLogUI : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← ExeptionLogUI

Fields

m_hiddenObjects

```
[SerializeField]  
private GameObject[] m_hiddenObjects
```

Field Value

GameObject[]

m_logContents

```
[SerializeField]  
private GameObject m_logContents
```

Field Value

GameObject

m_logDisplayed

```
private bool m_logDisplayed
```

Field Value

bool ↗

Methods

LogDisplay()

```
public void LogDisplay()
```

Class HidePanel

Namespace: [SamuraiSoccer.UI](#)

Assembly: SamuraiSoccer.UI.dll

巻物の右側をHideパネルで隠しておく UIのレンダー順はいじれなかったのでスクリプト側で解決した

```
public class HidePanel : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← HidePanel

Fields

m_parent

```
[SerializeField]  
private RectTransform m_parent
```

Field Value

RectTransform

m_pos

```
[SerializeField]  
private Vector3 m_pos
```

Field Value

Vector3

Methods

Start()

```
private void Start()
```

Update()

```
public void Update()
```

Class OpponentNumber

Namespace: [SamuraiSoccer.UI](#)

Assembly: SamuraiSoccer.UI.dll

```
public class OpponentNumber : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← OpponentNumber

Fields

m_easyCPUManager

```
[SerializeField]  
private EasyCPUManager m_easyCPUManager
```

Field Value

[EasyCPUManager](#)

m_remainNumberText

```
[SerializeField]  
private Text m_remainNumberText
```

Field Value

Text

Methods

Start()

```
private void Start()
```

Class PauseButton

Namespace: [SamuraiSoccer.UI](#)

Assembly: SamuraiSoccer.UI.dll

```
public class PauseButton : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← PauseButton

Fields

m_enablePause

```
[SerializeField]  
private bool m_enablePause
```

Field Value

[bool](#)

m_pausePanel

```
[SerializeField]  
private GameObject m_pausePanel
```

Field Value

GameObject

Methods

ContinueButton()

"ゲームに戻る"ボタンを押したときの処理

```
public void ContinueButton()
```

MenuBackButton()

"メニューに戻る"ボタンを押したときの処理

```
public void MenuBackButton()
```

OnClick()

```
public void OnClick()
```

RestartButton()

"ゲームをやり直す"ボタンを押したときの処理

```
public void RestartButton()
```

Start()

```
private void Start()
```

Class ScrollScript

Namespace: [SamuraiSoccer.UI](#)

Assembly: SamuraiSoccer.UI.dll

巻物をなめらかに移動させる

```
public class ScrollScript : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← ScrollScript

Fields

FlagMaterials

```
[SerializeField]  
private Material[] FlagMaterials
```

Field Value

Material[]

ScrollMaterial

```
private MeshRenderer ScrollMaterial
```

Field Value

MeshRenderer

ScrollObject

```
[SerializeField]  
private GameObject ScrollObject
```

Field Value

GameObject

initRot

```
private Vector3 initRot
```

Field Value

Vector3

rectra

```
[SerializeField]  
private RectTransform rectra
```

Field Value

RectTransform

rotSpeed

```
[SerializeField]  
private float rotSpeed
```

Field Value

[float](#)

slideTime

```
[SerializeField]  
private float slideTime
```

Field Value

[float](#)

Methods

ChangeMaterial(Stage)

巻物のMaterialを変更する

```
public void ChangeMaterial(Stage nowStage)
```

Parameters

[nowStage Stage](#)

巻物に反映させる国

ResetObject(float)

巻物を元初期位置に戻す

```
public void ResetObject(float startX)
```

Parameters

[startX float](#)

巻物の初期位置

ScrollSlide(float, float, float, float, CancellationToken)

巻物の回転処理を行う

```
public UniTask ScrollSlide(float startX, float goalX, float Y, float rollTime,  
CancellationToken cancellationToken = default)
```

Parameters

startX [float](#)

巻物の初期位置のX座標

goalX [float](#)

巻物の最終位置のX座標

Y [float](#)

巻物のY座標

rollTime [float](#)

移動、回転にかける時間

cancellationToken [CancellationToken](#)

Returns

UniTask

Start()

```
public void Start()
```

easeOutCubic(float, float, float, float)

イージングの計算

```
private float easeOutCubic(float t, float goal, float start, float goalTime)
```

Parameters

t `float`

現在の時刻

goal `float`

最終地点のx座標

start `float`

$t=0$ のx座標

goalTime `float`

最終地点の時刻

Returns

`float`

現在のx座標

Class SelectedStagePublisher

Namespace: [SamuraiSoccer.UI](#)

Assembly: SamuraiSoccer.UI.dll

```
public class SelectedStagePublisher : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← SelectedStagePublisher

Fields

m_stage

```
[SerializeField]  
[Tooltip("選択したステージの国がどこかを設定")]  
private Stage m_stage
```

Field Value

[Stage](#)

Methods

OnClick()

選択したステージとともに状態遷移を発行

```
public void OnClick()
```

Class SlidePad

Namespace: [SamuraiSoccer.UI](#)

Assembly: SamuraiSoccer.UI.dll

```
public class SlidePad : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← SlidePad

Fields

m_fingerID

```
private int m_fingerID
```

Field Value

[int](#)

m_isDragged

```
private bool m_isDragged
```

Field Value

[bool](#)

m_joyStartPosition

```
private Vector2 m_joyStartPosition
```

Field Value

Vector2

m_joyrect

```
private RectTransform m_joyrect
```

Field Value

RectTransform

m_joystick

```
[SerializeField]  
private GameObject m_joystick
```

Field Value

GameObject

m_radius

```
private float m_radius
```

Field Value

[float](#) ↗

m_scale

```
private float m_scale
```

Field Value

[float](#) ↗

m_slideStartPosition

```
private Vector2 m_slideStartPosition
```

Field Value

Vector2

Methods

Controller(Vector2)

```
private void Controller(Vector2 dir)
```

Parameters

dir Vector2

DragEnd()

おそらくEventTriggerで呼びだす タッチ終了検出

```
public void DragEnd()
```

DragStart(BaseEventData)

おそらくEventTriggerで呼びだす 最初のタッチの検出

```
public void DragStart(BaseEventData baseEventData)
```

Parameters

baseEventData BaseEventData

FindFinger()

タッチ情報から前フレームと同じタッチがあればそれを返す 初回タッチは別のところで登録するのでreturn new Touch();は呼ばれない

```
private Touch FindFinger()
```

Returns

Touch

LateUpdate()

```
private void LateUpdate()
```

PlayingState()

```
private void PlayingState()
```

Start()

```
private void Start()
```

Class TargetWorldData

Namespace: [SamuraiSoccer.UI](#)

Assembly: SamuraiSoccer.StageContents.StageSelect.dll

```
public class TargetWorldData : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← TargetWorldData

Fields

m_cameraDistance

```
[SerializeField]  
private float m_cameraDistance
```

Field Value

[float](#)

m_stageNumber

```
[SerializeField]  
private int m_stageNumber
```

Field Value

[int](#)

Properties

CameraDistance

```
public float CameraDistance { get; private set; }
```

Property Value

[float](#)

StageNumber

```
public int StageNumber { get; private set; }
```

Property Value

[int](#)

Methods

Start()

```
private void Start()
```

Class Test_ChargeAttackButton

Namespace: [SamuraiSoccer.UI](#)

Assembly: SamuraiSoccer.UI.dll

```
public class Test_ChargeAttackButton : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← Test_ChargeAttackButton

Methods

Start()

```
private void Start()
```

Update()

```
private void Update()
```

Class TextScroller

Namespace: [SamuraiSoccer.UI](#)

Assembly: SamuraiSoccer.UI.dll

UIテキストの一文字ずつ送る動作を行う。

```
public class TextScroller : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← TextScroller

Fields

m_fullText

```
private string m_fullText
```

Field Value

[string](#)

m_guardTime

```
[SerializeField]  
private int m_guardTime
```

Field Value

[int](#)

m_isTouched

```
private bool m_isTouched
```

Field Value

[bool](#) ↗

m_provider

```
[SerializeField]  
private TouchProvider m_provider
```

Field Value

[TouchProvider](#)

m_text

```
[SerializeField]  
private Text m_text
```

Field Value

Text

m_touchGuard

```
private bool m_touchGuard
```

Field Value

[bool](#) ↗

m_waitms4char

```
[SerializeField]  
private int m_waitms4char
```

Field Value

[int](#)

Methods

GuardTimer()

```
private UniTask GuardTimer()
```

Returns

UniTask

InitializeState(string)

```
private void InitializeState(string text)
```

Parameters

[text](#) [string](#)

OnTouched(bool)

```
private void OnTouched(bool isTouch)
```

Parameters

[isTouch](#) [bool](#)

ShowText(string)

文字を一文字ずつ表示する。

```
public UniTask ShowText(string text)
```

Parameters

text string ↗

表示する文字.

Returns

UniTask

Start()

```
private void Start()
```

Update()

```
private void Update()
```

Class TimerScript

Namespace: [SamuraiSoccer.UI](#)

Assembly: SamuraiSoccer.SoccerGame.dll

```
public class TimerScript : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← TimerScript

Fields

m_elapsedTime

```
private float m_elapsedTime
```

Field Value

[float](#)

m_end

```
private bool m_end
```

Field Value

[bool](#)

m_isPlaying

```
private bool m_isPlaying
```

Field Value

[bool](#)

m_limitTimeSec

```
[SerializeField]
private float m_limitTimeSec
```

Field Value

[float](#)

m_longWhistleNum

```
[SerializeField]
private int m_longWhistleNum
```

Field Value

[int](#)

m_resultSceneName

```
[SerializeField]
private string m_resultSceneName
```

Field Value

[string](#)

m_timeText

```
[SerializeField]
private Text m_timeText
```

Field Value

Text

Methods

FinishProcess()

```
private UniTask FinishProcess()
```

Returns

UniTask

Start()

```
private void Start()
```

Update()

```
private void Update()
```

Class TouchProvider

Namespace: [SamuraiSoccer.UI](#)

Assembly: SamuraiSoccer.UI.dll

```
public class TouchProvider : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← TouchProvider

Fields

m_reactiveProperty

```
private ReactiveProperty<bool> m_reactiveProperty
```

Field Value

ReactiveProperty<[bool](#)>

Properties

IsTouchingReactiveProperty

```
public IReadOnlyReactiveProperty<bool> IsTouchingReactiveProperty { get; }
```

Property Value

IReadOnlyReactiveProperty<[bool](#)>

Methods

Awake()

```
private void Awake()
```

Update()

```
private void Update()
```

Class UIFade

Namespace: [SamuraiSoccer.UI](#)

Assembly: SamuraiSoccer.UI.dll

```
public class UIFade : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← UIFade

Fields

m_fadeSpeed

```
[SerializeField]  
private float m_fadeSpeed
```

Field Value

[float](#)

m_images

```
[SerializeField]  
[Tooltip("フェードしたいUI")]  
private List<Image> m_images
```

Field Value

[List](#) <Image>

m_isStartEnabled

```
[SerializeField]
private bool m_isStartEnabled
```

Field Value

[bool](#)

Methods

FadeInUI()

画像のα値を増やすことで画像を出現させる

```
public UniTask FadeInUI()
```

Returns

UniTask

FadeOutUI()

画像のα値を減らすことで画像を見えないようにする

```
public UniTask FadeOutUI()
```

Returns

UniTask

Start()

```
private void Start()
```

Namespace SamuraiSoccer.UK

Classes

[PanjanExplode](#)

[PanjanMake](#)

[PanjanRoll](#)

Class PanjanExplode

Namespace: [SamuraiSoccer.UK](#)

Assembly: SamuraiSoccer.StageContents.UK.dll

```
public class PanjanExplode : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← PanjanExplode

Fields

fire

```
private GameObject fire
```

Field Value

GameObject

isBurn

```
private bool isBurn
```

Field Value

[bool](#)

m_calledNum

```
private int m_calledNum
```

Field Value

[int](#)

m_resultSceneName

```
[SerializeField]  
private string m_resultSceneName
```

Field Value

[string](#)

panjanMake

```
private PanjanMake panjanMake
```

Field Value

[PanjanMake](#)

Methods

SetFireObject(GameObject)

```
public void SetFireObject(GameObject fire)
```

Parameters

fire GameObject

Start()

```
private void Start()
```

Class PanjanMake

Namespace: [SamuraiSoccer.UK](#)

Assembly: SamuraiSoccer.StageContents.UK.dll

```
public class PanjanMake : MonoBehaviour
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← PanjanMake

Fields

fire

```
[SerializeField]  
private GameObject fire
```

Field Value

GameObject

isEnd

```
private bool isEnd
```

Field Value

[bool](#)

panjan

```
[SerializeField]  
private GameObject panjan
```

Field Value

GameObject

panjanExist

```
private bool panjanExist
```

Field Value

bool ↗

player

```
[SerializeField]  
private Transform player
```

Field Value

Transform

quaternion

```
private Quaternion quaternion
```

Field Value

Quaternion

respone

```
private Vector3 respone
```

Field Value

Vector3

Methods

Start()

```
private void Start()
```

Class PanjanRoll

Namespace: [SamuraiSoccer.UK](#)

Assembly: SamuraiSoccer.StageContents.UK.dll

```
public class PanjanRoll : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← PanjanRoll

Fields

capsuleCollider

```
[SerializeField]  
private CapsuleCollider capsuleCollider
```

Field Value

CapsuleCollider

exploded

```
private bool exploded
```

Field Value

[bool](#)

fire

```
private GameObject fire
```

Field Value

GameObject

moveSpeed

```
[SerializeField]  
private float moveSpeed
```

Field Value

[float](#)

panjanExplode

```
private PanjanExplode panjanExplode
```

Field Value

[PanjanExplode](#)

partMax

```
[SerializeField]  
private int partMax
```

Field Value

[int](#)

player

```
private Transform player
```

Field Value

Transform

playing

```
private bool playing
```

Field Value

bool ↗

rb

```
[SerializeField]  
private Rigidbody rb
```

Field Value

Rigidbody

rot

```
[SerializeField]  
private GameObject rot
```

Field Value

GameObject

rotSpeed

```
[SerializeField]  
private float rotSpeed
```

Field Value

[float](#)

soundEffect

```
[SerializeField]  
private AudioSource soundEffect
```

Field Value

Methods

Explode()

```
private void Explode()
```

SetObjects(GameObject, Transform)

```
public void SetObjects(GameObject fire, Transform player)
```

Parameters

fire GameObject

player Transform

Start()

```
private void Start()
```

Update()

```
private void Update()
```

selfExplode()

```
private void selfExplode()
```

Namespace Tutorial

Classes

[Tutorial](#)

Class Tutorial

Namespace: [Tutorial](#)

Assembly: SamuraiSoccer.StageContents.Tutorial.dll

```
public class Tutorial : MonoBehaviour
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← Tutorial

Fields

arrowAnimator

```
public Animator arrowAnimator
```

Field Value

Animator

ball

```
public GameObject ball
```

Field Value

GameObject

chargeAttackText

```
public GameObject chargeAttackText
```

Field Value

GameObject

chargeAttackTutorialFinished

```
private bool chargeAttackTutorialFinished
```

Field Value

[bool ↗](#)

easyCPUManager

```
public EasyCPUManager easyCPUManager
```

Field Value

[EasyCPUManager](#)

enemy

```
public GameObject enemy
```

Field Value

GameObject

enemyGroup

```
public GameObject enemyGroup
```

Field Value

GameObject

enemyNumber

```
public Text enemyNumber
```

Field Value

Text

exclamationMark

```
public GameObject exclamationMark
```

Field Value

GameObject

initBallPos

```
private Vector3 initBallPos
```

Field Value

Vector3

isThreeOnThreeFinished

```
private bool isThreeOnThreeFinished
```

Field Value

[bool](#) ↗

leftControllerFocusAnimator

```
public Animator leftControllerFocusAnimator
```

Field Value

Animator

referee

```
public GameObject referee
```

Field Value

GameObject

remainChargeAttackNum

```
public int remainChargeAttackNum
```

Field Value

[int](#)

remainChargeAttackText

```
public Text remainChargeAttackText
```

Field Value

Text

rightControllerFocusAnimator

```
public Animator rightControllerFocusAnimator
```

Field Value

Animator

samurai

```
public GameObject samurai
```

Field Value

GameObject

samuraiCamera

```
public CinemachineVirtualCamera samuraiCamera
```

Field Value

CinemachineVirtualCamera

spotCamera

```
public CinemachineVirtualCamera spotCamera
```

Field Value

CinemachineVirtualCamera

teamGroup

```
public GameObject teamGroup
```

Field Value

GameObject

textAnimator

```
public Animator textAnimator
```

Field Value

Animator

tutorialText

```
public Text tutorialText
```

Field Value

Text

uiMask

```
public GameObject uiMask
```

Field Value

GameObject

whistleSENumber

```
[Tooltip("ホイッスル開始音")]
public int whistleSENumber
```

Field Value

[int↗](#)

wholeviewCamera

```
public CinemachineVirtualCamera wholeviewCamera
```

Field Value

CinemachineVirtualCamera

Methods

Awake()

```
private void Awake()
```

ChargeAttack(CancellationToken)

```
private UniTask ChargeAttack(CancellationToken cancellation_token = default)
```

Parameters

cancellation_token [CancellationToken↗](#)

Returns

UniTask

CheckChargeAttackTutorialFinished()

```
public void CheckChargeAttackTutorialFinished()
```

CheckThreeOnThreeCleared()

```
private bool CheckThreeOnThreeCleared()
```

Returns

[bool](#)

MinusChargeAttackNum()

```
public void MinusChargeAttackNum()
```

Opening(CancellationToken)

```
private UniTask Opening(CancellationToken cancellation_token = default)
```

Parameters

[cancellation_token](#) [CancellationToken](#)

Returns

UniTask

RetryThreeOnThree(CancellationToken)

```
private UniTask RetryThreeOnThree(CancellationToken cancellation_token = default)
```

Parameters

`cancellation_token` [CancellationToken](#)

Returns

UniTask

Runner(CancellationToken)

```
private UniTask Runner(CancellationToken cancellation_token = default)
```

Parameters

`cancellation_token` [CancellationToken](#)

Returns

UniTask

Start()

```
private void Start()
```

ThreeOnThree(CancellationToken)

```
private UniTask ThreeOnThree(CancellationToken cancellation_token = default)
```

Parameters

`cancellation_token` [CancellationToken](#)

Returns

UniTask

UIDescription(CancellationToken)

```
private UniTask UIDescription(CancellationToken cancellation_token = default)
```

Parameters

cancellation_token [CancellationToken](#)

Returns

UniTask

Namespace UnityEngine.UI

Classes

[LetterSpacing](#)

Class LetterSpacing

Namespace: [UnityEngine.UI](#)

Assembly: SamuraiSoccer.UI.dll

```
[AddComponentMenu("UI/Effects/Letter Spacing", 15)]  
public class LetterSpacing : BaseMeshEffect, IMeshModifier
```

Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← UIBehaviour ← BaseMeshEffect ← LetterSpacing

Implements

IMeshModifier

Inherited Members

BaseMeshEffect.OnEnable() , BaseMeshEffect.OnDisable() ,
BaseMeshEffect.OnDidApplyAnimationProperties() , BaseMeshEffect.OnValidate() ,
BaseMeshEffect.ModifyMesh(Mesh) , BaseMeshEffect.graphic , UIBehaviour.Awake() , UIBehaviour.Start() ,
UIBehaviour.OnDestroy() , UIBehaviour.IsActive() , UIBehaviour.Reset() ,
UIBehaviour.OnRectTransformDimensionsChange() , UIBehaviour.OnBeforeTransformParentChanged() ,
UIBehaviour.OnTransformParentChanged() , UIBehaviour.OnCanvasGroupChanged() ,
UIBehaviour.OnCanvasHierarchyChanged() , UIBehaviour.IsDestroyed()

Constructors

LetterSpacing()

```
protected LetterSpacing()
```

Fields

SupportedTagRegexPattersn

```
private const string SupportedTagRegexPattersn = "<b>|</b>|<i>|</i>|<size=.*?>|</size>|  
<color=.*?>|</color>|<material=.*?>|</material>"
```

Field Value

[string](#) ↗

m_spacing

```
[SerializeField]
private float m_spacing
```

Field Value

[float](#) ↗

useRichText

```
[SerializeField]
private bool useRichText
```

Field Value

[bool](#) ↗

Properties

spacing

```
public float spacing { get; set; }
```

Property Value

[float](#) ↗

Methods

GetRegexMatchedTagCollection(string, out int)

```
private IEnumerator GetRegexMatchedTagCollection(string line, out int lineLengthWithoutTags)
```

Parameters

line [string](#)

lineLengthWithoutTags [int](#)

Returns

[IEnumerator](#)

ModifyMesh(VertexHelper)

```
public override void ModifyMesh(VertexHelper vh)
```

Parameters

vh VertexHelper

ModifyVertices(List<UIVertex>)

```
public void ModifyVertices(List<UIVertex> verts)
```

Parameters

verts [List](#)<UIVertex>