Universal Turn Based AI

1.0

Generated by Doxygen 1.8.9.1

Thu Feb 26 2015 13:33:02

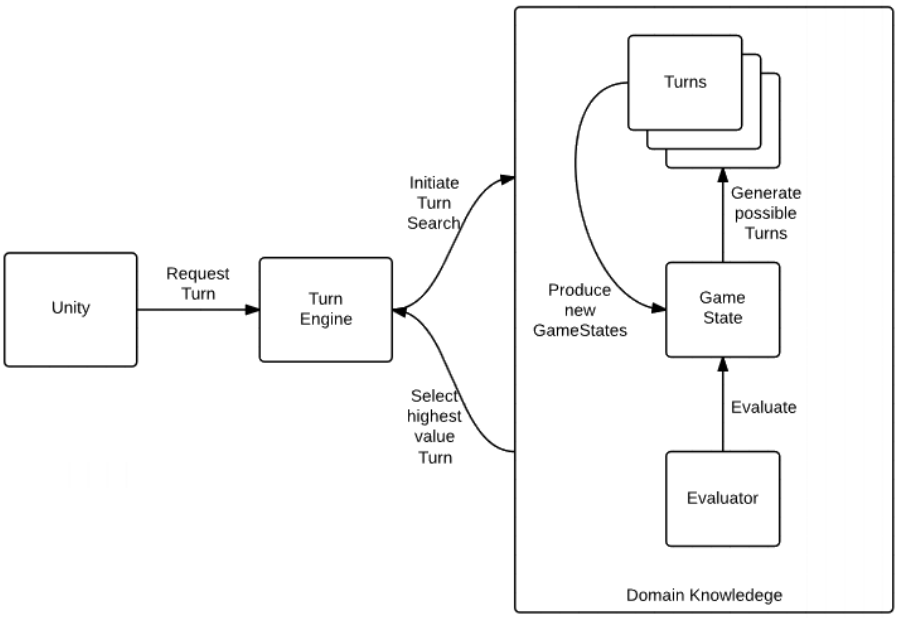
Contents

Chapter 1

Universal Turn Based AI

1.1 What is Universal Turn Based AI?

Universal Turn Based AI is a general purpose solution for implementing an artificial intelligence player for any turn based game in Unity. It is essentially a generic implementation of the Minimax algorithm with Alpha-Beta pruning that can be applied to any game domain. Both a single threaded and multi-threaded implementation are provided. All the searching and move selection is done internally by the TurnEngine, so you must simply provide implementations of IGameState, ITurn and IEvaluator. This will provide the system with your particular game's domain knowledge, informing the TurnEngine on how to represent, generate and evaluate game states.



This diagram shows the general structure of the Universal Turn Based AI system. Requests for the next move to be made come in from the left into the TurnEngine. A request is just the current IGameState which should contain information about which player's turn is next. The request is processed and a turn search is set up to run in a separate thread.

The TurnEngine is preconfigured at initialisation to search for a set amount of time, to a certain number of moves ahead or both. Each IGameState must be capable of generating each ITurn that is possible and valid for that game state and each ITurn must be capable of converting its starting game state to a new valid game state. This creates the turn search loop.

Each new game state is evaluated using an IEvaluator implementation, also provided to the turn engine at initialisation. The evaluator computes the value of a given game state in terms of how close that game state is to a winning game state for a particular player. This allows the turn engine to determine which initial moves are better or worse by looking some number of moves ahead and assuming optimal play by each player where optimal play is defined as: "provides the best value from the IEvaluator implementation".

1.2 Multi-Threaded vs. Single Threaded

Whilst this system does provide a multi-threaded TurnEngine implementation it may not perform better than the single threaded implementation for your particular game. This is due to the high overhead of managing multiple threads. The multi-threaded implementation was included to give users the option to see if it provides better results for their game's domain. It typically out performs the single threaded implementation for game domains where the search tree is extremely wide i.e. at each game state there are VERY many possible moves that can be made.

Chapter 2

Namespace Index

2.1 Namespace List

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

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

Namespace Documentation

5.1 Package UniversalTurnBasedAI

Classes

- class EngineStats

  Used to collect stastics from the engine
- class EvaluatorRandom

  An Evaluator that returns random values for every state. Can be useful to test other evaluation functions. Any evaluation function should be at least as good as selecting moves randomly.
- interface IEvaluator

  An Evaluator defines an evaluation function to determine the value of a IGameState from the point of view of a particular player.
- interface IGameState

  Represents the current state of some game. Implement this interface to provide a TurnEngine with domain specific knowledge. The effiency of IsTerminal, GeneratePossibleTurns and Clone will all effect the performance of the TurnEngine.
- interface ITurn

  Represents the sum of actions that a player can take on their turn in the game.
- class MinimaxWorker

  Holds all of the initialising information required for a Minimax search so it can more easily be passed to a ThreadPool
- class TurnEngine

  The super-class for all Turn Engines. Implementations of this class control the search for the best ITurn. Provides an entry point for Unity with GetNextTurn which can be used in a familiar coroutine pattern. Defines attributes common to all Turn Engines such as depth and time limits. Also provides the TurnReadyEvent which is triggered after a turn search has been completed and returns the best turn found.
- class TurnEngineMultiThreaded

  A multi-threaded implementation of TurnEngine. Uses the same search algorithm as TurnEngineSingleThreaded but runs each initial branch in a separate thread.
- class TurnEngineSingleThreaded

  A single threaded implementation of TurnEngine. Uses an implementation of the Minimax algorithm with Alpha-Beta pruning.

Chapter 6

Class Documentation

6.1 UniversalTurnBasedAI.EngineStats Class Reference

Used to collect stastics from the engine

Public Member Functions

- override string **ToString** ()

Properties

- float **AverageDepth** `[get]`
- float **AverageTime** `[get]`

6.1.1 Detailed Description

Used to collect stastics from the engine

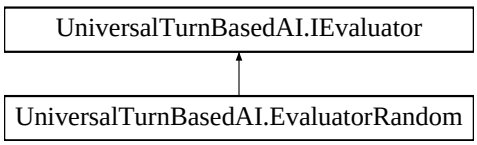
The documentation for this class was generated from the following file:

- Assets/UniversalTurnBasedAI/Core/EngineStats.cs

6.2 UniversalTurnBasedAI.EvaluatorRandom Class Reference

An Evaluator that returns random values for every state. Can be useful to test other evaluation functions. Any evaluation function should be at least as good as selecting moves randomly.

Inheritance diagram for UniversalTurnBasedAI.EvaluatorRandom:



Public Member Functions

- EvaluatorRandom (float min, float max)

Initializes a new instance of the UniversalTurnBasedAI.EvaluatorRandom class.

- float GetMinValue ()

  Gets the minimum value any state can have

- float GetMaxValue ()

  Gets the max value any state can have

- float Evaluate (IGameState state)

  Evaluate the specified GameState. Good evaluation functions should return GetMaxValue on a winning state and GetMinValue on a losing state. This method must also provide value to non-terminal states that give the engine some indication of whether the player is closer or further away from winning.

6.2.1 Detailed Description

An Evaluator that returns random values for every state. Can be useful to test other evaluation functions. Any evaluation function should be at least as good as selecting moves randomly.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 UniversalTurnBasedAI.EvaluatorRandom.EvaluatorRandom (float *min*, float *max*) [inline]

Initializes a new instance of the UniversalTurnBasedAI.EvaluatorRandom class.

Parameters

min	The minimum value to generate
max	The maximum value to generate

6.2.3 Member Function Documentation

6.2.3.1 float UniversalTurnBasedAI.EvaluatorRandom.Evaluate (IGameState *state*) [inline]

Evaluate the specified GameState. Good evaluation functions should return GetMaxValue on a winning state and GetMinValue on a losing state. This method must also provide value to non-terminal states that give the engine some indication of whether the player is closer or further away from winning.

As this will need to be called on every searched IGameState the efficiency of this method is directly related to the performance of a TurnEngine.

Parameters

state	The state to evaluate

Implements UniversalTurnBasedAI.IEvaluator.

6.2.3.2 float UniversalTurnBasedAI.EvaluatorRandom.GetMaxValue ( ) [inline]

Gets the max value any state can have

Returns

The max value.

Implements UniversalTurnBasedAI.IEvaluator.

6.2.3.3 float UniversalTurnBasedAI.EvaluatorRandom.GetMinValue ( ) [inline]

Gets the minimum value any state can have

Returns

The minimum value.

Implements UniversalTurnBasedAI.IEvaluator.

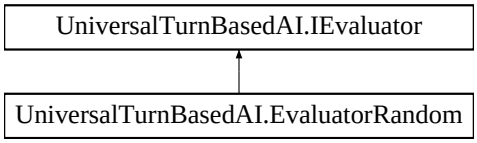
The documentation for this class was generated from the following file:

- Assets/UniversalTurnBasedAI/Core/EvaluatorRandom.cs

6.3 UniversalTurnBasedAI.IEvaluator Interface Reference

An Evaluator defines an evaluation function to determine the value of a IGameState from the point of view of a particular player.

Inheritance diagram for UniversalTurnBasedAI.IEvaluator:



Public Member Functions

- float GetMinValue ()

  Gets the minimum value any state can have
- float GetMaxValue ()

  Gets the max value any state can have
- float Evaluate (IGameState state)

  Evaluate the specified GameState. Good evaluation functions should return GetMaxValue on a winning state and GetMinValue on a losing state. This method must also provide value to non-terminal states that give the engine some indication of whether the player is closer or further away from winning.

6.3.1 Detailed Description

An Evaluator defines an evaluation function to determine the value of a IGameState from the point of view of a particular player.

NOTE: Evaluators are not cloned while searching so try to design your evaluation functions such that they do not need to track information or keep an internal state as it will get overwritten when searching other nodes. If want to track information do so inside the IGameState implementation as these are cloned on generation and ensure that your state information is cloned as well.

See also

TurnEngine, IGameState, ITurn

6.3.2 Member Function Documentation

6.3.2.1 float UniversalTurnBasedAI.IEvaluator.Evaluate (IGameState *state*)

Evaluate the specified GameState. Good evaluation functions should return GetMaxValue on a winning state and GetMinValue on a losing state. This method must also provide value to non-terminal states that give the engine some indication of whether the player is closer or further away from winning.

As this will need to be called on every searched IGameState the efficiency of this method is directly related to the performance of a TurnEngine.

Parameters

state	The state to evaluate

Implemented in UniversalTurnBasedAI.EvaluatorRandom.

6.3.2.2 float UniversalTurnBasedAI.IEvaluator.GetMaxValue ( )

Gets the max value any state can have

Returns

The max value.

Implemented in UniversalTurnBasedAI.EvaluatorRandom.

6.3.2.3 float UniversalTurnBasedAI.IEvaluator.GetMinValue ( )

Gets the minimum value any state can have

Returns

The minimum value.

Implemented in UniversalTurnBasedAI.EvaluatorRandom.

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

- Assets/UniversalTurnBasedAI/Core/IEvaluator.cs

6.4 UniversalTurnBasedAI.IGameState Interface Reference

Represents the current state of some game. Implement this interface to provide a TurnEngine with domain specific knowledge. The effiency of IsTerminal, GeneratePossibleTurns and Clone will all effect the performance of the TurnEngine.

Public Member Functions

- bool IsTerminal ()

  Determines whether this GameState is terminal.
- IEnumerable< ITurn > GeneratePossibleTurns ()

  A coroutine for generating every possible ITurn. Generating turns this way allows turns to be generated and search lazily. The efficiency of generating each turn effects the performance of the TurnEngine. You will only get the benefits of lazy evaluation if you yield each turn after generating them. Generating all turns and then iterating through them will negatively effect the performance of the TurnEngine.
- IGameState Clone ()

  Returns an exact copy of the current GameState with new references. Any GameState information that is altered by a ITurn should be deep copied to prevent Turns writing over shared information.

6.4.1 Detailed Description

Represents the current state of some game. Implement this interface to provide a TurnEngine with domain specific knowledge. The effiency of IsTerminal, GeneratePossibleTurns and Clone will all effect the performance of the TurnEngine.

Some information that is typically useful is: whose turn it is and some representation of the state of each component of the game. It can also be useful to track some meta-information about the state that can be used by an evaluation function rather than having the evaluation function compute the information every time. For example, counts of things within the GameState.

See also

TurnEngine, ITurn, IEvaluator

6.4.2 Member Function Documentation

6.4.2.1 IGameState UniversalTurnBasedAI.IGameState.Clone ( )

Returns an exact copy of the current GameState with new references. Any GameState information that is altered by a ITurn should be deep copied to prevent Turns writing over shared information.

The efficiency of this method effects the performance of the TurnEngine, try to copy as little information as possible but remember that you will probably need new instances of any reference types that can be altered by a ITurn.

6.4.2.2 IEnumerable<ITurn> UniversalTurnBasedAI.IGameState.GeneratePossibleTurns ( )

A coroutine for generating every possible ITurn. Generating turns this way allows turns to be generated and search lazily. The efficiency of generating each turn effects the performance of the TurnEngine. You will only get the benefits of lazy evaluation if you yield each turn after generating them. Generating all turns and then iterating through them will negatively effect the performance of the TurnEngine.

This method should only produce turns that are valid for this particular GameState

Returns

The possible turns.

6.4.2.3 bool UniversalTurnBasedAI.IGameState.IsTerminal ( )

Determines whether this GameState is terminal.

Returns

`true` If this GameState is terminal; otherwise, `false`.

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

- Assets/UniversalTurnBasedAI/Core/IGameState.cs

6.5 UniversalTurnBasedAI.ITurn Interface Reference

Represents the sum of actions that a player can take on their turn in the game.

Public Member Functions

- IGameState ApplyTurn (IGameState state)

  Applies this ITurn to state giving the resulting IGameState. The TurnEngine clones state before passing it to this function when called internally to prevent the original GameState from being overridden

  See also

  IGameState, TurnEngine

6.5.1 Detailed Description

Represents the sum of actions that a player can take on their turn in the game.

See also

IGameState, TurnEngine

6.5.2 Member Function Documentation

6.5.2.1 IGameState UniversalTurnBasedAI.ITurn.ApplyTurn (IGameState *state*)

Applies this ITurn to *state* giving the resulting IGameState. The TurnEngine clones *state* before passing it to this function when called internally to prevent the original GameState from being overridden

See also

IGameState, TurnEngine

Returns

The GameState that is a result of applying this turn to *state* .

Parameters

state	The state to apply the turn to.

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

- Assets/UniversalTurnBasedAI/Core/ITurn.cs

6.6 UniversalTurnBasedAI.MinimaxWorker Class Reference

Holds all of the initialising information required for a Minimax search so it can more easily be passed to a ThreadPool

Public Member Functions

- MinimaxWorker (IGameState rootState, ITurn firstTurn, IEvaluator eval, int maxDepth, bool ourTurn, EventWaitHandle waitHandle)

  Initializes a new instance of the UniversalTurnBasedAI.MinimaxWorker class.
- void **EvaluateState** (object threadState)
- float **AlphaBeta** (IGameState state, IEvaluator eval, int depth, float alpha, float beta, bool ourTurn)
- void **Stop** ()

Public Attributes

- ITurn firstTurn

  The first turn to use in the search this is the turn retrieved for returning

Properties

- float Value `[get]`

  The value of firstTurn

6.6.1 Detailed Description

Holds all of the initialising information required for a Minimax search so it can more easily be passed to a ThreadPool

6.6.2 Constructor & Destructor Documentation

6.6.2.1 UniversalTurnBasedAI.MinimaxWorker.MinimaxWorker (IGameState *rootState,* ITurn *firstTurn,* IEvaluator *eval,* int *maxDepth,* bool *ourTurn,* EventWaitHandle *waitHandle*) [inline]

Initializes a new instance of the UniversalTurnBasedAI.MinimaxWorker class.

Parameters

rootState	The starting state
firstTurn	The turn to apply to the starting state to generate this worker's branch
eval	The Evaluator
maxDepth	Max depth.
ourTurn	Whether or not it is the searching player's turn
waitHandle	Signals the ThreadPool that the search is complete

6.6.3 Member Data Documentation

6.6.3.1 ITurn UniversalTurnBasedAI.MinimaxWorker.firstTurn

The first turn to use in the search this is the turn retrieved for returning

6.6.4 Property Documentation

6.6.4.1 float UniversalTurnBasedAI.MinimaxWorker.Value [get]

The value of *firstTurn*

The value.

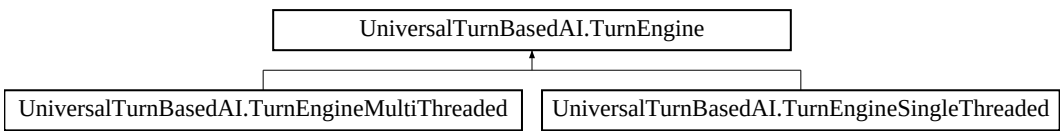
The documentation for this class was generated from the following file:

- Assets/UniversalTurnBasedAI/Core/MinimaxWorker.cs

6.7 UniversalTurnBasedAI.TurnEngine Class Reference

The super-class for all Turn Engines. Implementations of this class control the search for the best ITurn. Provides an entry point for Unity with GetNextTurn which can be used in a familiar coroutine pattern. Defines attributes common to all Turn Engines such as depth and time limits. Also provides the TurnReadyEvent which is triggered after a turn search has been completed and returns the best turn found.

Inheritance diagram for UniversalTurnBasedAI.TurnEngine:



Public Member Functions

- delegate void **TurnReady** (ITurn bestTurn)
- System.Collections.IEnumerator GetNextTurn (IGameState state)

  The entry point to the engine. Starts a new thread to run the search in and waits on it. Once the search is completed or timed out, calls the TurnReadyEvent to return the best turn found.
- EngineStats **ResetStatisticsLog** ()
- virtual void **Stop** ()

Protected Member Functions

- void InitEngine (IEvaluator eval, float timeLimit, int depthLimit, bool timeLimited, bool collectStats)

  Initialises the common engine elements.
- void ExecuteAndCatch (Action< object > action, object arg, Action< Exception > exceptionHandler)

  Wrapper for catching exceptions from another thread
- void ExceptionHandler (Exception ex)

  Writes .NET exceptions to the Unity Debug Log
- abstract void **TurnSearchDelegate** (object state)

Static Protected Member Functions

- static T **GetRandomElement**< **T** > (IList< T > list)

Protected Attributes

- int **maxDepth**
- float **maxTime**
- bool **timeLimited** = false
- IEvaluator **eval**
- System.Random **rando**
- DateTime **startTime**
- bool **collectStats** = false
- bool **stopped** = true
- ITurn **bestTurn**

Properties

- EngineStats Stats `[get]`

  Property for accessing stats if there were collect.
- bool **Exit** `[get]`

Events

- TurnReady TurnReadyEvent

  Triggered after GetNextTurn has been called and the found turn is ready to be returned. bestTurn will be the best turn discovered by the engine

6.7.1 Detailed Description

The super-class for all Turn Engines. Implementations of this class control the search for the best ITurn. Provides an entry point for Unity with GetNextTurn which can be used in a familiar coroutine pattern. Defines attributes common to all Turn Engines such as depth and time limits. Also provides the TurnReadyEvent which is triggered after a turn search has been completed and returns the best turn found.

See also

IGameState, ITurn, IEvaluator

6.7.2 Member Function Documentation

6.7.2.1 void UniversalTurnBasedAI.TurnEngine.ExceptionHandler (Exception *ex*) [inline], [protected]

Writes .NET exceptions to the Unity Debug Log

Parameters

ex	The exception

6.7.2.2 void UniversalTurnBasedAI.TurnEngine.ExecuteAndCatch (Action< object > *action,* object *arg,* Action< Exception > *exceptionHandler*) [inline], [protected]

Wrapper for catching exceptions from another thread

Parameters

action	The action to run
arg	The action's arguments
exceptionHandler	A handler for the exceptions

6.7.2.3 System.Collections.IEnumerator UniversalTurnBasedAI.TurnEngine.GetNextTurn (IGameState *state*) [inline]

The entry point to the engine. Starts a new thread to run the search in and waits on it. Once the search is completed or timed out, calls the TurnReadyEvent to return the best turn found.

Typical usage from Unity is:

```
StartCoroutine(engine.GetNextTurn(state));
```

Parameters

state	State.

6.7.2.4 void UniversalTurnBasedAI.TurnEngine.InitEngine (IEvaluator *eval,* float *timeLimit,* int *depthLimit,* bool *timeLimited,* bool *collectStats*) [inline], [protected]

Initialises the common engine elements.

Parameters

eval	The class used to evaluate GameStates searched by this engine
timeLimit	The maximum time allowed for search, in seconds. Must be greater than 0
depthLimit	The maximum depth to search in the GameState search tree. Also called "ply". Must be at least 1
timeLimited	If set to `true` Search will end after the set timeLimit, otherwise search will complete to the set depthLimit
collectStats	If set to `true` collect statistics.

6.7.3 Property Documentation

6.7.3.1 EngineStats UniversalTurnBasedAI.TurnEngine.Stats [get]

Property for accessing stats if there were collect.

If stat collection enabled: returns any collected statisics collected since the last ResetStatisticsLog call otherwise returns a new, empty EngineStats

6.7.4 Event Documentation

6.7.4.1 TurnReady UniversalTurnBasedAI.TurnEngine.TurnReadyEvent

Triggered after GetNextTurn has been called and the found turn is ready to be returned. bestTurn will be the best turn discovered by the engine

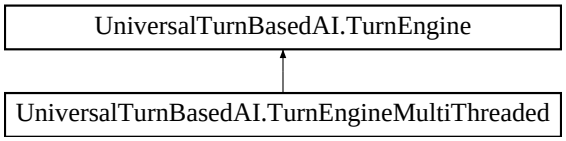
The documentation for this class was generated from the following file:

- Assets/UniversalTurnBasedAI/Core/TurnEngine.cs

6.8 UniversalTurnBasedAI.TurnEngineMultiThreaded Class Reference

A multi-threaded implementation of TurnEngine. Uses the same search algorithm as TurnEngineSingleThreaded but runs each initial branch in a separate thread.

Inheritance diagram for UniversalTurnBasedAI.TurnEngineMultiThreaded:



Public Member Functions

- TurnEngineMultiThreaded (IEvaluator eval, float timeLimit, bool collectStats)

  Initializes a new instance of the UniversalTurnBasedAI.TurnEngineMultiThreaded class with a time limit. Once the time limit has been reached the best turn found so far will be bestTurn
- TurnEngineMultiThreaded (IEvaluator eval, int depthLimit, bool collectStats)

  Initializes a new instance of the UniversalTurnBasedAI.TurnEngineMultiThreaded class with a depth limit. Searches the entire GameState tree up to the specific depth. The best turn found after the search will be bestTurn .
- TurnEngineMultiThreaded (IEvaluator eval, float timeLimit, int depthLimit, bool collectStats)

  Initializes a new instance of the UniversalTurnBasedAI.TurnEngineMultiThreaded class.
- override void **Stop** ()

Protected Member Functions

- override void TurnSearchDelegate (object state)

  A wrapper for the Minimax algorithm. Initialises the first branch of turns so that they can be given values and the best possible returned. Always generates at least one possible turns so that at least some sensible result can be returned. When the search is completed or timed out bestTurn will be assigned to the best found turn.

Additional Inherited Members

6.8.1 Detailed Description

A multi-threaded implementation of TurnEngine. Uses the same search algorithm as TurnEngineSingleThreaded but runs each initial branch in a separate thread.

This implementation may not be significantly faster than using TurnEngineSingleThreaded due to the overhead of managing multiple threads. May see an improvement if your GameState search tree is extremely wide i.e. in each state there is a very large number of possible moves to make.

See also

TurnEngine, TurnEngineSingleThreaded

6.8.2 Constructor & Destructor Documentation

6.8.2.1 UniversalTurnBasedAI.TurnEngineMultiThreaded.TurnEngineMultiThreaded (IEvaluator *eval,* float *timeLimit,* bool *collectStats*) [inline]

Initializes a new instance of the UniversalTurnBasedAI.TurnEngineMultiThreaded class with a time limit. Once the time limit has been reached the best turn found so far will be *bestTurn*

Parameters

eval	The Evaluator
timeLimit	The time limit, must be greater than 0
collectStats	If set to `true` collect stats.

6.8.2.2 UniversalTurnBasedAI.TurnEngineMultiThreaded.TurnEngineMultiThreaded (IEvaluator *eval,* int *depthLimit,* bool *collectStats*) [inline]

Initializes a new instance of the UniversalTurnBasedAI.TurnEngineMultiThreaded class with a depth limit. Searches the entire GameState tree up to the specific depth. The best turn found after the search will be *bestTurn* .

Parameters

eval	Eval.
depthLimit	Depth limit, must be at least 1
collectStats	If set to `true` collect stats.

6.8.2.3 UniversalTurnBasedAI.TurnEngineMultiThreaded.TurnEngineMultiThreaded (IEvaluator *eval,* float *timeLimit,* int *depthLimit,* bool *collectStats*) [inline]

Initializes a new instance of the UniversalTurnBasedAI.TurnEngineMultiThreaded class.

Parameters

eval	The IEvaluator to use
timeLimit	Time limit in seconds. Must be greater than 0
depthLimit	Depth limit or maximum "ply". Must be at least 1
collectStats	If set to `true` collect stats.

6.8.3 Member Function Documentation

6.8.3.1 override void UniversalTurnBasedAI.TurnEngineMultiThreaded.TurnSearchDelegate (object *state*) [inline], [protected], [virtual]

A wrapper for the Minimax algorithm. Initialises the first branch of turns so that they can be given values and the best possible returned. Always generates at least one possible turns so that at least some sensible result can be returned. When the search is completed or timed out bestTurn will be assigned to the best found turn.

For each initial turn this created a new MinimaxWorker is created and added to a ThreadPool. Then it waits for each thread to finish or a time out.

See also

MinimaxWorker

Parameters

state	The starting state

Implements UniversalTurnBasedAI.TurnEngine.

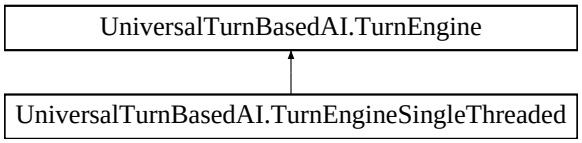
The documentation for this class was generated from the following file:

- Assets/UniversalTurnBasedAI/Core/TurnEngineMultiThreaded.cs

6.9 UniversalTurnBasedAI.TurnEngineSingleThreaded Class Reference

A single threaded implementation of TurnEngine. Uses an implementation of the Minimax algorithm with Alpha-Beta pruning.

Inheritance diagram for UniversalTurnBasedAI.TurnEngineSingleThreaded:



Public Member Functions

- TurnEngineSingleThreaded (IEvaluator eval, float timeLimit, bool collectStats)

  Initializes a new instance of the UniversalTurnBasedAI.TurnEngineSingleThreaded class with a time limit. Once the time limit has been reached the best turn found so far will be bestTurn
- TurnEngineSingleThreaded (IEvaluator eval, int depthLimit, bool collectStats)

  Initializes a new instance of the UniversalTurnBasedAI.TurnEngineSingleThreaded class with a depth limit. Searches the entire GameState tree up to the specific depth. The best turn found after the search will be bestTurn .
- TurnEngineSingleThreaded (IEvaluator eval, float timeLimit, int depthLimit, bool collectStats)

  Initializes a new instance of the UniversalTurnBasedAI.TurnEngineSingleThreaded class with both a time and depth limit.

Protected Member Functions

- override void TurnSearchDelegate (object state)

  A wrapper for the Minimax algorithm. Initialises the first branch of turns so that they can be given values and the best possible returned. Always generates at least one possible turns so that at least some sensible result can be returned. When the search is completed or timed out bestTurn will be assigned to the best found turn.

Additional Inherited Members

6.9.1 Detailed Description

A single threaded implementation of TurnEngine. Uses an implementation of the Minimax algorithm with Alpha-Beta pruning.

See also

TurnEngine, TurnEngineMultiThreaded

6.9.2 Constructor & Destructor Documentation

6.9.2.1 UniversalTurnBasedAI.TurnEngineSingleThreaded.TurnEngineSingleThreaded (IEvaluator *eval,* float *timeLimit,* bool *collectStats*) [inline]

Initializes a new instance of the UniversalTurnBasedAI.TurnEngineSingleThreaded class with a time limit. Once the time limit has been reached the best turn found so far will be *bestTurn*

Parameters

eval	The Evaluator
timeLimit	The time limit, must be greater than 0
collectStats	If set to `true` collect stats.

6.9.2.2 UniversalTurnBasedAI.TurnEngineSingleThreaded.TurnEngineSingleThreaded (IEvaluator *eval,* int *depthLimit,* bool *collectStats*) [inline]

Initializes a new instance of the UniversalTurnBasedAI.TurnEngineSingleThreaded class with a depth limit. Searches the entire GameState tree up to the specific depth. The best turn found after the search will be *bestTurn* .

Parameters

eval	Eval.
depthLimit	Depth limit, must be at least 1
collectStats	If set to `true` collect stats.

6.9.2.3 UniversalTurnBasedAI.TurnEngineSingleThreaded.TurnEngineSingleThreaded (IEvaluator *eval,* float *timeLimit,* int *depthLimit,* bool *collectStats*) [inline]

Initializes a new instance of the UniversalTurnBasedAI.TurnEngineSingleThreaded class with both a time and depth limit.

Parameters

eval	The IEvaluator to use
timeLimit	Time limit in seconds. Must be greater than 0
depthLimit	Depth limit or maximum "ply". Must be at least 1
collectStats	If set to `true` collect stats.

6.9.3 Member Function Documentation

6.9.3.1 override void UniversalTurnBasedAI.TurnEngineSingleThreaded.TurnSearchDelegate (object *state*) [inline], [protected], [virtual]

A wrapper for the Minimax algorithm. Initialises the first branch of turns so that they can be given values and the best possible returned. Always generates at least one possible turns so that at least some sensible result can be returned. When the search is completed or timed out bestTurn will be assigned to the best found turn.

Parameters

state	The starting state

Implements UniversalTurnBasedAI.TurnEngine.

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

- Assets/UniversalTurnBasedAI/Core/TurnEngineSingleThreaded.cs

Index