

Automated Commentaries for Simulated Soccer

CommentScheduler Class

Audience	All
Author	Justin Hogg
Scope	
Date Created	05/03/2007
Version Number	0.1
Version History	0.1 – Original Document (JH)
Reviewed	Yes
Last saved by Justin Hogg, 18/03/2007	

Sign-off sheet

Date: 18/03/2007

Document Author Signature: JH*

Document Author Name: Justin Hogg

Quality Assurance Signature: AS*

Quality Assurance Name: Akbar Sherwani

Project Manager Signature: AM*

Project Manager Name: Ahsan Mussa

* By signing this document you approve that the entire contents of the deliverable has been reviewed and is in line with the objectives of the project.

Automated Commentaries for Simulated Soccer

1. Introduction

This class is responsible for the prioritisation and scheduling of all marked up comments received from the CommentaryProducer class.

1.1 Overview

As the GameAnalyser (GA) class receives updates from the Soccer Server every 100ms, it is not possible to output (as audio) every possible game event – note that although these updates are received every 100ms, the same game event may take place over a number of these 100ms periods e.g. a player may dribble the ball for several seconds.

During busy game periods involving a fast succession of different events, it is not possible to output audio commentary relating to every game event. This is because although an event may take place very quickly in real time, the audio comment describing that event may be up to three seconds in duration. Whilst this comment is spoken, several other events may occur. Once the current audio output for a comment is finished, there is a negative impact on the system if it produces audio output for all events that occurred during the period of the previous audio comment. If this is done, a noticeable lag will occur, that is, the action taking place in real time on the soccer monitor is ahead of the audio that describes it.

For the most effective commentary, the audio must relate to the current event taking place on the soccer monitor i.e. be in real time – or perhaps very close where this is not possible, and especially if the events have a very high importance such as a goal.

This is the purpose of the CS class – to receive comments representing all game events, and to determine the best possible selection for audio output, based on importance and real time synchronization with the game events on the soccer monitor.

1.2 CommentaryScheduler (CS)

It was necessary to create this class to encapsulate the objects, methods and variables used by the system for prioritisation, into a single class. This would give clarity over the previous attempt at prioritisation that was coded into the CP class. As the CP class was designed for marking up and producing comments for each type of game event, adding prioritisation to the class violated good object orientation principles and made the code harder to read and debug.

The CS class is coded as a subclass of Thread, and as a result it executes in a separate thread to the main program. This allows it to perform tasks concurrently with other threads in the system, and to poll continuously to determine the exact time the next comment is to be spoken.

Automated Commentaries for Simulated Soccer

The CS class contains a proposition pool – this is the set of all marked up (with variables) comments relating to events that have occurred in the game over a three second period.

While an audio comment is output, the proposition pool fills with marked up comments relating to game events occurring within the duration of that current audio comment. After three seconds – the length of the longest comment – the CS class selects the most suitable comment from the pool to output as audio, and empties the proposition pool. The three second period that starts when any comment begins its audio output, ensures that any audio comment will have finished before the next one begins, and prevents the voice classes having to deal with any prioritisation or scheduling issues, and also ensures the system functions correctly given that the voice classes do not offer a buffer or queue for comments – if several comments are passed to the voice classes within a three second period, then the latter comments will overwrite any previous ones within the same three second period, and they will not be output.

At the end of a three second period, it is not enough to simply output the highest comment in the proposition pool, as it may have occurred, in extreme cases, 1 or 2 seconds ago. If this comment is then output as audio, it will not be synchronized with the game event on the soccer monitor, as the event took place in the past, and the game state will have progressed and changed. This leads to audio commentary that describes events that have already taken place, and are not the current state of play on the soccer monitor.

For this reason, at the end of a three second period, the CS thread not only selects the comment with the highest importance rating (1 is the highest importance, 12 is the lowest), but if it has an importance rating of twelve, then it also tests if that comment was added to the pool less than 100ms ago. This is the optimal period to ensure audio comments are coordinated with the game events taking place on the soccer monitor, whilst allowing enough comments to be spoken. Comments with an importance rating of 1-11 are spoken regardless of the time they were added, as they must be spoken and a slight lag in audio is preferable over an important event not being output at all. Comments with a rating of 1-11 are spaced in time sufficiently to prevent a high number being present in the proposition pool at once in any given three second period, and this ensures that any lag is minimal. Extensive testing proves this to be the case.

If the 100ms test is reduced to 50ms then but a few comments are spoken, this is due to the time overhead of the code executing. If the period is increased, then the audio commentary no longer runs in real time – it falls behind the events taking place on the soccer monitor.

Two threads access the proposition pool so access to the pool is synchronised to eliminate race hazards. The first thread is the main program thread, and this makes method calls from the CP class to the CS class to add marked up comments to the proposition pool. The second thread is the thread dedicated to the CS class – the thread that polls on the proposition pool,

Automated Commentaries for Simulated Soccer

selects the next comment to be output, and then empties the pool. Only one thread can operate on the shared resource at a time (the proposition pool) to prevent erroneous behaviour e.g. the main thread adds a comment to the pool after the CS thread has selected the next audio comment, but before it empties the pool. This new comment could then be deleted with the remaining contents of the pool, and the system would not consider it at the end of the next three-second period – a critical condition if it was a goal or similarly important event. Synchronization ensures only one thread can access the pool at a time; any others must wait until the first has finished its activity on the shared resource and has released it. This can lead to a slight performance increase (in time) due to any blocked threads having to wait until the shared resource is released – this is a contributory factor to the required 100ms period described above.

Comments with a rating of 12 are optional and can be discarded if necessary. This increases efficiency and real time synchronization between the game and the commentary. Most frequent game events are rated 12 including passes, loss of possession, dribbling, ball possession lost and pressure on the defence. If a comment being added to proposition pool is of a lower importance than one already in the pool, then it is discarded – there is no point adding it to the pool, as the higher priority comment will be selected at the end of the three-second period. The lock on the shared resource should be the shortest amount of time possible, as the next comment to be spoken cannot be selected (by the separate CS thread) until the shared resource is available – this will be longer if all comments are added to the pool.

The CS thread continuously polls on the proposition pool and tests if it has been three seconds since the last audio comment. This increases response time if three seconds has elapsed but no comment is available that was added within the last 100ms. By continually polling, the thread will detect a new comment immediately and output it to the speech engine.

The system offers two different voices, one for game events and another for statistical comments. A Boolean parameter identifies which voice is to output a given comment. This variable is included (and set to true) in method calls made by the CS to the voice classes, if the statistical voice is to output a comment. If this parameter is not provided, the voice defaults to the standard game event voice.

2. Previous versions

Before the final version described above, the CS class thread slept for three seconds after iteration through the main run () methods loop. This was fine if a comment had been passed to the speech engine. If it hadn't, it would sleep again for three seconds and this heavily increased the silent periods between audio comments, if no comment was available.

It wasn't until the final version, that a test was introduced to ensure that comments with an importance rating of 12 had been added to the pool in the

Automated Commentaries for Simulated Soccer

last 100ms. This, in conjunction with the sleep issue described in the previous paragraph, led to the commentary becoming out of sync with the real time game events. The CS thread would wake and output the highest importance comment, regardless of its age, which could be a second or two.

Before the introduction of the CS class, voice excitement levels had previously been determined in the CP class as this had output comments to the voice classes. Upon introduction of the CS class, the CP method calls no longer output the audio. Instead, the CP calls added each comment to the proposition pool in the CS class. It became the function of the CS class to select the best comment and pass it to the voice classes for audio output. In the final version, this includes selecting a voice excitement level for each comment, depending on its importance.

3. Evaluation

The final version increases CPU usage to around 70% due to polling being used instead of a thread sleep mechanism. CPU usage has already been deemed out of scope for our project, unless it has a direct impact on our system, and this is not the case. It is certainly worth the trade off as the system performs better with a much tighter real time synchronization of the commentary and the game.

The synchronization of the proposition pool does lead to a delay in processing the waiting thread as it blocks. The class has been engineered so this has minimal impact on the system - comments are only added to the pool if they have a higher importance than the current highest importance comment in the pool – reducing the blocking time where possible (This is not the case for comments with an importance rating of 12, but this ensures when an optional comment is spoken, it is concurrent with the event taking place on the screen.). The synchronization of the proposition pool is absolutely necessary to prevent race hazards associated with a multi thread environment.

If a comment is passed to the voice classes for audio output, it cannot be cancelled. This can lead to a delay of an important event being output as audio – if a pass is output, and a goal occurs 100ms seconds later, the system will not produce audio commentary relating to the goal until approximately 2900ms later. This is a serious lag but still preferable over the goal being mentioned. In practice (determined by hours of watching games for testing purposes) this is never the case. A goal would occur more than 100ms after a pass as the player receiving the ball would need time to react, and the ball would take more time to reach the goal. Testing has shown that goals are spoken immediately as the ball crosses the line into a goal, or very soon afterwards.

Goals and shots are the most important events suffering a slight lag. Other events that may suffer a slight lag have no effect on the real time commentary as they relate to time events or game wins/draws. It does not have the same

Automated Commentaries for Simulated Soccer

implications if there is a 1 second delay before the commentary announces which team won the game, or that the game is nearly over.

The system does not provide audio commentary for every pass or frequent event of low importance. As a conscious design choice, the group decided to implement a richer commentary to differentiate our system from the systems that formed the subject of our research. These systems mention most passes or possessions with a short phrase. Our system offers a richer commentary with longer, more informative phrases. Our system identifies long passes, for example, by adding a phrase to a comment to identify this. A pass gives details of both the receiver and the passer; so two possessions of the ball (by different players) can be relayed to the listener in a single comment. Goals give details of the player who scored, the distance of the shot and the total number of goals for that player. These are amongst other comments that relay a more informative view of the game, not just the current event. This works very well and adds a great sense of realism to the commentary.

Longer, informative comment phrases lead to more events occurring during the output of a single (longer) comment. If the system attempted to output audio for every game event, a serious real time lag would ensue. This has been identified in our design, and an optimum balance has been achieved between maintaining a real time synchronization between the audio commentary and the events taking place on the soccer monitor, and producing commentary on enough game events that a listener is able to follow the flow of the game.

However, as a future development, when a comment is passed to the voice classes for audio output, instead of waiting three seconds before passing the next comment, the length of the comment could be identified. If the comment is shorter than three seconds, the next comment could be output as audio the instant the previous one has finished. This would eliminate the silent periods at the end of comments that are less than three seconds, and allow more comments to be output. This would lead to a commentary that was almost continuous and non-lifelike even though it could cover more game events.

Also, as a further development, the CS could have an awareness of ball position on the field, and proximity to players. This could be used to determine if a comment in the proposition pool represents the current game state. Such functionality would also be very useful in deciding when to output statistical comments. If the ball was being passed around the central area of the football pitch, statistical comments could be output that offer information on player performance, attack patterns, previous game trends, and more.

The system is stable and offers a robust implementation. Comments are synchronised with the game events occurring on the soccer monitor so the commentary is easy to follow. Commentary is also varied and informative, adding realism and a flavour that supplements the visual action on the soccer monitor.