**NURBS Path Finding Algorithm for RTS (Real Time Strategy Games)**

Synopsis

The NURBS/Bezier Path Finding Algorithm is an algorithm that uses implicit curves to find the ideal path through an obstacle course.

Due to the structure of maps in a game it is more often that if the player finds one path that path will continue and if one finds a mountains or water, they are likely to keep encountering them for a significant while.

The key to the algorithm is, in comparison to other sequential search algorithms, that in the majority of paths, there exists large stretches of continuous space which does not alter course, gaining an advantage in these stretches similar to way RLE (run-length encoding) takes advantage of long strings of identical color.

Reiterating, most path planning algorithms which will work on short stretch checking only a few blocks for collision and then onwards, repeating this process. This can be computationally intensive. This algorithm takes advantage of the stretches of space that occur in maps by representing them implicitly using a NURBs curve. The NURBS Curve represents the path and is begins as a straight line from start to finish). Also, note that other space search algorithms can easily be integrated into this algorithm and may just use this as supplemental.

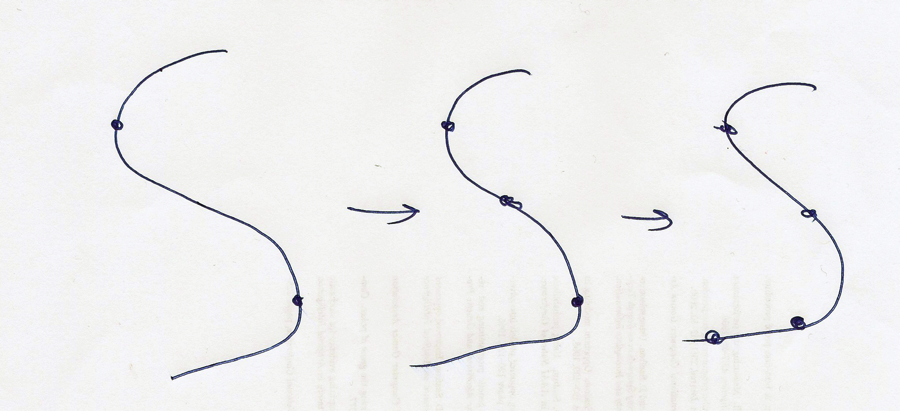
Overview

(Initialization)

The algorithm begins by creating a NURBS curve that consists of a straight line from the current location (current node) to the end location (goal node). Along the NURBS curve (in this instance a line), a random number of weight locations are assigned. These weight nodes are multiplied over time, increasing as a clear path is not found.

(Maintenance)

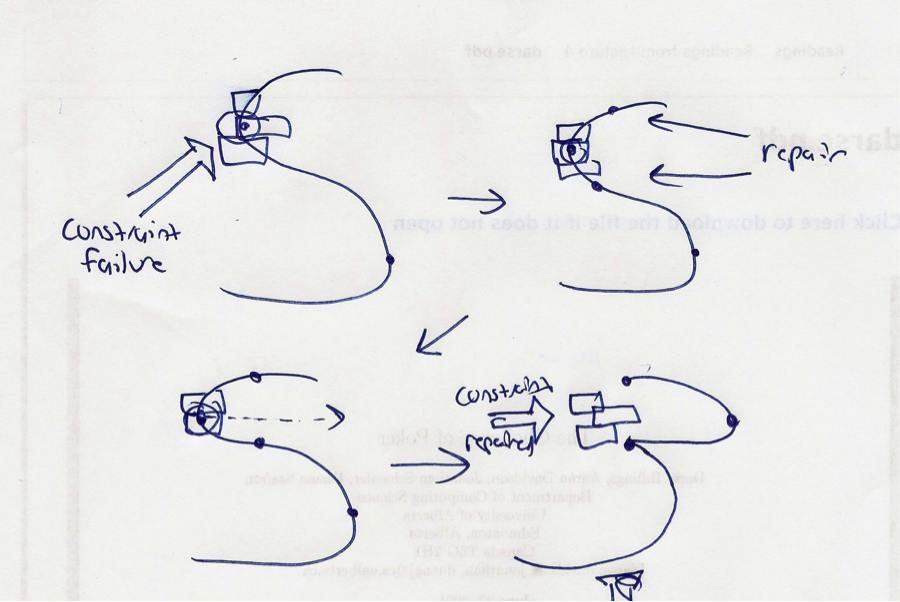
**Figure 1.1**

****

**Fig 1.1**

**Iteration Maintenance**

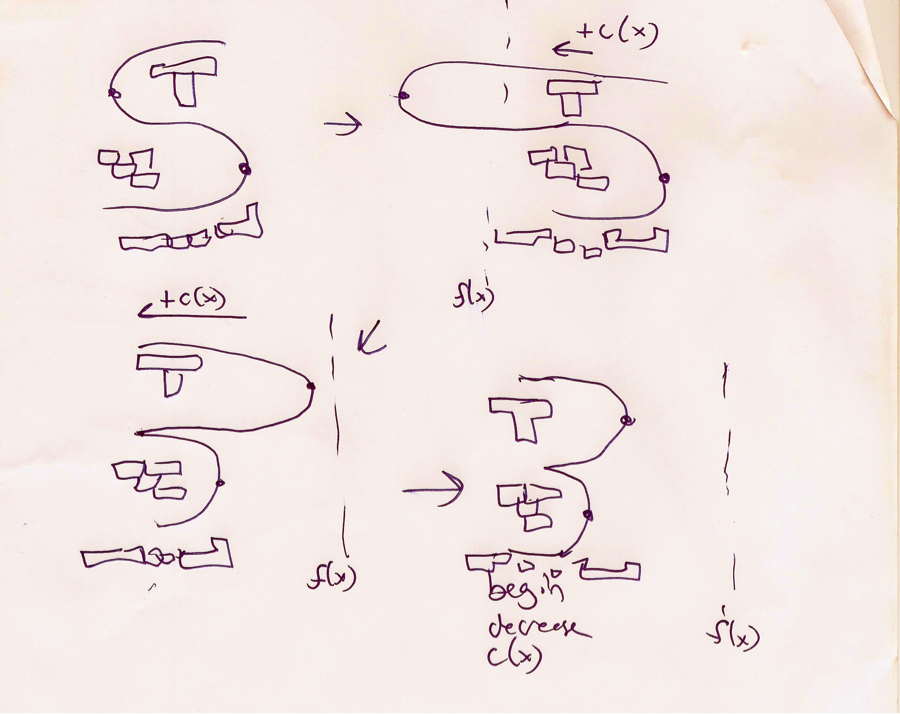
The algorithm goes through a limited number of iterations, during each iteration the algorithm will increase the number of weights assigned to the curve according to a given heuristic. The assignment of the weights to the curve has the constraint that the weights are always to be placed in traversable locations. (See Fig 1.1)



**Fig 1.2**

**Constraint Failure**

During iteration, if the weight fails the constraint that it is placed in a traversable location, a binary division occurs at the proposed location, with a weight placed before and a weight placed after the proposed location. Therefore, in this instance, three weights are placed instead of one. (See Fig 1.2) Note: We do not want to cross one individual path.



**Fig 1.3**

**Weight Strengthening**

During the main loop body, in each iteration the algorithm will slightly increase the strength of each weight according to a given constant **c** or a heuristic **c(x)**. When the strength reaches the flex-maximum, that is, the threshold for the curve strength, where going beyond the flex-maximum for a heuristic **f(x),** reaches a strongly unshapely shape for the curve (no path would likely ever look like the given curve), the curve will reverse its polarity (e.g., instead of curving up will curve down or vice versa) and thus begin decreasing curvature in this new polarity until the loop completes. (see Fig 1.3)

(Determining Loop Termination)

The loop termination process first considers the location of the weights and makes certain no weights are in a constraint violating location. With this general test,

In parallel, each stretch of the path from weight to weight performs a divide & conquer procedure, validating that that is a clear path to travel. Also, note, that for static environments or partially static environments, not the player, but the environment data found in the course of the algorithm may be cached for future use by the same agent or by a new agent.

(Loop Termination)

When the curve is not blocked (there are no constraint failures for this iteration), the loop will perform a “Partial Full Patch Check”, the weight intersection test in the paragraph above. A “full path check” will then be performed. The full path check will step through the Bezier curve path to verify the complete path. A property of the Bezier curve path is that if it does not fail in any constraints, a correct path that can be taken has been found (requires formal proof). If the full path check returns true, the complete path has been found.