Dynamic Strategies in a Real-Time Strategy Game

William Joseph Falke II and Peter Ross

School of Computing, Napier University
10 Colinton Road, Edinburgh EH10 5DT, UK
joe@bongofury.vispa.com, P.Ross@napier.ac.uk

Abstract. Most modern real-time strategy computer games have a sophisticated but fixed 'AI' component that controls the computer's actions. Once the user has learned how such a game will react, the game quickly loses its appeal. This paper describes an example of how a learning classifier system (based on Wilson's ZCS [1]) can be used to equip the computer with dynamically-changing strategies that respond to the user's strategies, thus greatly extending the games playability for serious gamers.

1 The Game and the Classifier System

Real-time strategy (RTS) games typically involve fighting a battle against a computer opponent. This opponent typically has a very sophisticated but essentially static strategy and once the user has learned how it behaves, the game loses its playability. To get round this limitation we have used a learning classifier system (LCS) to provide the computer with the capability to dynamically change its strategy. LCSs are sometimes criticized for being slow to learn. In this example, the LCS has short conditions so as to fit in the real-time loop; success also depends on a careful choice of the types of actions.

The game we implemented, using the publicly available version of the Auran Jet game engine, involves two opposing armies each consisting of two squads of 20 soldiers, moving on hilly terrain. A squad defaults to moving in a 6-6-6-2 formation but the soldiers are individually governed by a flocking algorithm [2] so that squad shapes are very fluid in battles. The essence of the game play is to jockey for position, because it pays for a squad to attack the flank of the enemy or for two squads to attack a third if its companion squad is too far away to offer support quickly enough. Figure 1 illustrates the idea. The rectangles are identifying battle-flags. The large diamond is the cursor that the user employs to command his squads, by placing the cursor somewhere and then left- or right-clicking to summon a particular squad towards that location.

The classifier system sends commands to the computer's squads, at randomly-chosen times within a two- to four-second interval (experimentally determined to offer good playability). The condition part has 14 bits as follows, the values depend on the squad to be commanded. For enemy squad 1, two bits describe how far away it is; one bit indicates whether that squad is engaged in a fight;

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2724, pp. 1920–1921, 2003.

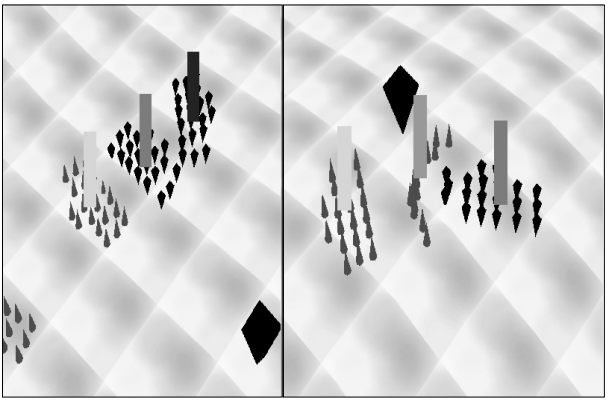

Fig. 1. Two examples of attacks

one bit indicates whether that squad is stronger than this one; one bit indicates whether this squad is positionally flanking that one. Another five bits give the same data about enemy squad 2. Bits 11 and 12 give the proximity of the friendly squad, bits 13 and 14 show whether this squad and its friend are currently engaged in battle. A battle consists of a set of soldier-to-soldier fights running in parallel, each ending with the death of one combatant. A key to the success of the system is the fairly general nature of the possible actions [3]: pursue a named enemy squad, flank a named enemy squad, flee from a named enemy squad, follow the friendly squad. Unlike in the original LCS, the reward system is invoked immediately after the environment string is built and evaluates the squad's previous action. The reward is 1000 is that action caused more kills than deaths for the squad, 0 otherwise. In experiments it took around 750 computer-vs-computer battles, at 15 seconds per battle on a 750MHz PC, to generate a strong opponent for a human. Human testers then found the computer opponent to be both hard to beat and hard to model, as it continued learning.

This demonstrates by example that an LCS, kept suitably simple and with suitably general actions, can be the basis of an effective dynamic strategy in a real-time strategy game. The frame rate of 40-50 fps is good. Moreover, acquired strategies can be saved and exchanged with other users.

References

1. Wilson, S.: ZCS: a zeroth level classifier system. Evolutionary Computation **2** (1994) 1–18
2. Reynolds, C.: Flocks, herds, and schools: A distributed behavioural model. Computer Graphics **21** (1987) 25–34
3. Smith, R., Dike, B., Ravichandran, B., El-Fallah, A., Mehra, R.: The fighter aircraft lcs: a case of different lcs goals and techniques. In Lanzi, P.L., W.Stolzmann, Wilson, S., eds.: Learning Classifier Systems, From Foundations to Applications. Number 1813 in LNCS. Springer-Verlag (2000) 283–300