Taken from http://www.red3d.com/cwr/boids/RipOff_Flocking.html

Q: Which 1980 arcade game first demonstrated the animation technique now known as "flocking?"

A: "Rip-Off" was the flocking game. Also the first cooperative-play arcade game, it featured smart enemies that sought out target objects and could change goals while avoiding collisions with each other.

--- Trivia slide, SIGGRAPH 2001 Electronic Pre-show

Rip-Off: A Description of Programmed Behavior in a Video Game

Tim Skelly
Design Happy

This is not intended to be a history of the 1980 video arcade game "Rip-Off," which I designed and programmed. However, a little background will help explain why and how its computer-controlled opponents had an unusual amount of intelligence for the time..

By 1980 video arcade games like Pac-Man were beginning to show glimmers of today's "game AI," but their low-resolution displays limited movement of player and game icons to a very predictable left, right, up and down. Games with vector displays like Larry Rosenthal's Vectorbeam system displayed their linear graphics at high resolution, which allowed rotation and movement without image degradation. Unfortunately, low screen refresh rates limited vector games to virtually blank, black screens populated with small clumps of glowing lines. The only way for manufacturers of vector-based systems to compete was to create games that featured superior game play that only these displays could provide. One simple example to consider -- would Atari's "Asteroids" been possible, let alone popular, on an extremely low-resolution raster display? Not too likely.

The line-drawing limitations of video game vector displays meant that game tokens, icons and characters had to be small and few in number. Even using very short and very few lines, "Rip-Off" displayed only two player figures, three "enemy" figures and eight small triangles representing the objects to be "ripped off." Game scores were only displayed between waves of attackers.

The game play was this -- one or two players working together protected the eight "canisters" placed initially in the center of the screen. Up to three enemy "tanks" would enter from the edges of the screen and attempt to attach a canister to themselves, which they would then attempt to drag off the screen. If a player's tank shot and destroyed an escaping enemy, the canister would be left where it was until picked up by another enemy. The game ended when all eight objects had been removed from the screen. No instructions were given to the enemy tanks to avoid player tanks. On contact with any other vehicle, player or computer controlled, the player vehicles exploded and had to "wait offstage" for a second or two.

There were no "lives" as in other games. "Rip-Off" was Players VS Entropy.

Prior to "Rip-Off," Vectorbeam programmer Dan Sunday created the popular game "Tailgunner." In it, he introduced a tiny amount of "game AI" to save on memory. Rather than store the paths of enemy ships in detail, he chose to store only key points along those paths. Once activated, his space ships oriented themselves towards their target points as they traveled forward along their own main axis. This resulted in graceful, curved paths that could not be easily anticipated by the player. Those paths were, of course, spline curves..

Armed with this "orient and move" behavior, I was able to give my enemy tanks enough intelligence to complete their tasks. This is what I programmed them to do:

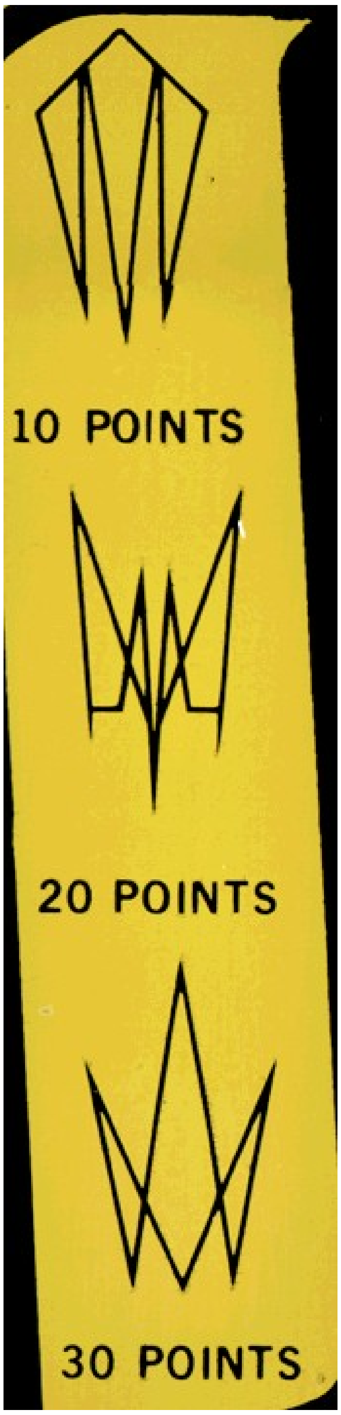
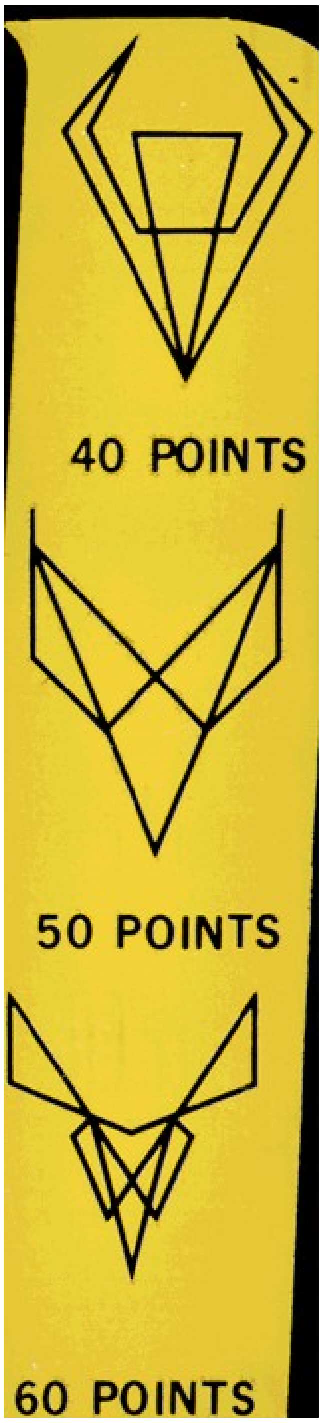
1. THIS IS WHAT YOU WANT: When initialized, the attacking tanks were each given a particular canister to retrieve. If there were more tanks than canisters, the extra tanks were put into "attack mode," which I will describe later.
2. GO GET IT: Each tank moved towards their assigned canister in the fashion of the ships in "Tailgunner." They moved forward while turning towards their target position, decelerating when near their targeted canister so as to prevent "orbiting."
3. HOOK IT TO YOUR TRAILER HITCH: When close enough to do so, an enemy tank would stop, rotate and attach the target canister to its tail end.
4. RUN FOR IT: Once attached to a canister, the tank was assigned a random exit point to accelerate towards. On reaching the exit, the tank and canister were removed from play.

So far, no significant departure from the seeking behavior used by Dan Sunday in "Tailgunner." But these enemies had more rules:

5. DON'T BUMP YOUR BUDDIES: When within a certain radius of another tank, each tank was to steer away from the other until out of range. (This was also used to initialize the tanks on entry. I placed each tank on the same spot and then let their avoidance A.I. and seeking behavior disperse them.)
6. GET AWAY FROM ME OR DIE: One behavior had priority over canister seeking and removal, but not avoidance. When within a certain radius of a player figure, if not attached to a canister, the enemies were programmed to make the player's tank their goal and to fire a short-range "beam." If that beam hit the player's tank, as in a collision, the player figure was removed from the screen for a few seconds, then placed back at its starting position. (Unlike the behavior when approaching a canister, when in attack mode the enemy ships did not decelerate when they neared their target. This meant that clever players could, and often did, allow enemy tanks to orbit them. This effectively put the game on "pause.")
7. IS THIS YOURS? TAKE MINE: As someone once said, the trick to making computers appear intelligent is simply to make them not do stupid things. An enemy tank that would drive right over an available canister on its way to the one assigned to it would look stupid. When within a certain radius of ANY canister not attached or in the process of being attached, enemy tanks were programmed to make the nearby canister their target canister and to go immediately into "pick-up" mode. Before doing so, the tank would check to see if the canister it was now attaching had been assigned to another tank. If so, it would swap canister assignments with that tank.

Content by Tim Skelly, August 23, 2001
HTML by Craig Reynolds, August 27, 2001

1. INSERT QUARTERS
   ADDITIONAL QUARTERS ADD CREDITS
2. PRESS START
   ONE PLAYER - ONE CREDIT
   TWO PLAYERS - TWO CREDITS
3. SCORE POINTS BY SHOOTING PIRATES
   PIRATE VALUES ADVANCE WHEN ONE FULL WAVE IS DESTROYED. TWO WAVES IN TWO PLAYER GAME.
   DESTROYING ALL SIX CLASSES OF PIRATES ADDS TEN POINTS TO BONUS LEVEL.
4. GAME IS OVER WHEN ALL FUEL CANISTERS HAVE BEEN RIPPED-OFF.


10 POINTS
20 POINTS
30 POINTS

40 POINTS
50 POINTS
60 POINTS