Review of Spaceship Combat Scenario Architecture

Samuel Roberts
University of Essex

November 4, 2013

1 Premise

The general premise of the experiments so far have been an extension on previous work relating to procedural design of spaceships within the context of a 2D physics-based game. While the last notable experiments relating to actual spaceship composition focused on simple movement, within this scenario the focus is instead on combat and survival. Spaceships are controlled by simple algorithmic controllers, based upon movement to a target point affected by weighted conditions and inputs. The intent of the experiment was such that an effective design and set of controller weights would be a more fit combination, and thus be the best end result of an evolutionary algorithm trial.

Initially, the experiment focused on the concept of two teams in opposition with each other, with the intent being the development of an arms race, where one team's designs shift to capitalise on weaknesses of the others. However, the two teams were treated as separate populations with evolutionary selection and propagation executed in isolation, and so did not actually have much capacity to respond to the other team designs. Using a single population and assigning ships to be in one team or the other did not produce the expected results either, and so the scenario was again changed to one where all spaceships are in competition with each other.

1.1 Scoring Spaceships

Scoring for spaceships occurs based on the result of a set of simulated combats, which trials the whole population, and a second design objective based series of rewards and penalties. The simulated combat involves all members of the population being initialised within the constraints of their chromosomes, and is based upon the spaceships moving around and shooting at other spaceships based on the controller output, within a finite walled environment that bounces spaceships that come into contact with its edges. Spaceships are scored at the end of a combat based on their performance.

A spaceship is rewarded for remaining hull, or health, points, very lightly penalised on number of bullets fired such that only large quantities of unnecessary bullets impact other fitness variables, and flatly penalised for being destroyed. With the result of one combat complete, four more combats are run and each spaceship receives the mean of their perfomance scores as their fitness score.

The final scoring occurs based on meeting design criteria. Spaceships are penalised for components being a distance greater than 50 pixels away from the origin of the spaceship based on how far over this limit the components drift, and are severely penalised for being too large to fit within the game world.

2 Spaceship Composition

Spaceships are composed of a point mass connected to a series of individual point components, which in turn define the vertices of a polygon that defines the shape of the spaceship. The relation of the chromosome to a spaceship is discussed in the Evolution section. Here, the parameters and components of a spaceship are defined as follows.

A spaceship possesses physical attributes, including a position and rotation, a velocity and an angular velocity, and a physical mass amount as well as moment of inertia. In addition to these physical properties, spaceships also have three finite quantities directly tied to the combat scenario. A spaceship has a limited amount of fuel, a limited amount of bullets, and a limited amount of hull. Fuel is used by thruster components in order to provide the spaceship with linear and rotational impulses for movement, bullets are used by turret components to fire at opposing spaceships, and hull is a measure of how many hits the spaceship can take before it is considered as destroyed.

2.1 Ship Components

The components that a spaceship can be made of are thrusters or turrets. Each of these components share in common their own mass and moment of inertia. When a spaceship is constructed, it is constructed such that each component is added sequentially. If a component is too close to where an existing component would be, it is skipped and not added. Adding a component increases the mass of the ship by the mass of the component as well as the mass of an invisible "strut" from the centre of the ship to the component, and the moment of inertia is increased by the moment of inertia of the component as well as by the same strut as mentioned, but with far less impact for moment of inertia than for mass.

After all components are added, their relative positions are altered slightly so that the geometrical centre and centre of mass for the spaceship are the same point.

2.1.1 Thruster

Thrusters are components that provide physical impulses of force to their parent ship, based upon their local position and rotation in relation to the ship's centre. They consume fuel equal to the amount of force exerted, and do not operate if the parent ship no longer has any fuel. Thrusters can be activated or deactivated, and consume fuel for every simulation tick they remain active.

2.1.2 Turret

Turrets are components that fire bullets based upon their local position and orientation. Bullets are consumed every time a turret fires, but every turret has an inherent "cooldown" period where no bullet can be fired. A turret fires a bullet when its cooldown period has elapsed and it is active. If the parent ship has no more bullets, the turret does not fire.

3 Other Game Entities

3.1 Projectiles

Projectiles are fired by turrets and have a finite time of existence within the game world. Collision with the edge of the world destroys them, running out of time to live destroys them and collision with a spaceship that is not their spaceship of origin destroys them, as well as inflicting damage upon the other ship. Bullets collide with spaceships based on the intersection of the point in space they occupy with the polygons of spaceships they are already intersecting the bounding circles of.

3.2 Pickups

Also existing within the game space, and placed at the same locations for every combat simulation, are pickups. These circular pickups collide based as a point intersecting the polygon of spaceships that overlap with them, same as bullets, but deliver some benefit. Three exist for each of the finite quantities possessed by a spaceship. Fuel pickups grant the spaceship more fuel, ammo pickups grant the spaceship more bullets, and hull pickups grant the spaceship more hull. No pickups can increase the amount of these quantities beyond the initial maximum universal to every spaceship.

4 Spaceship Controller

Given the set of ships, and access to all pickups and projectiles within the simulation, the controller focuses mostly on selection of targets relating to desired vectors of direction relating to some sub-objective. These direction vectors are weighted by parameters within the spaceship chromosome, such that individual ship controllers may have stronger weighting for some sub-objectives than others. Once all direction vectors have been calculated, normalised and scaled, they are summed to a final direction vector that is normalised, and the most appropriate thrust action that would move the ship in this direction is the thrust action taken.

These sub-objectives are firing upon enemy targets, chasing the mean of all enemies, approaching the nearest ally, avoiding bullets, and moving to collect pickups.

4.1 Calculating Directions

A target to fire at is acquired based on the closest potential target ship that is also predicted to be likely to be hit if fired upon at this exact moment in time. This hit prediction is based on a finite length hit-scan from all thrusters and assumes the target does not move. If no ships are likely to be hit, the target is instead chosen based solely on closest distance. The direction to the target is calculated and can be weighted such that a ship approaches the target or flees the target.

Additionally, a weighted direction is calculated to approach the mean position of all enemies, a condition that was originally more meaningful for the team-based scenario but remains for the purposes of allowing more interesting behaviour.

The closest ally is also selected purely on a distance-based basis, and the direction to the closest ally can also be weighted to encourage grouping or spreading out. With the simulation currently set up to be every spaceship operating for itself, this subcondition is mostly irrelevant and is in fact dummied out.

Bullet avoidance takes into consideration the direction of all projectiles that seem likely to collide with the ship, individually weighting them inversely based on distance. The closer a bullet is to a ship, the more important it becomes. All of these weighted directions are normalised into a single bullet avoidance direction vector, which can be weighted. Higher positive numbers would lead to the ship placing more priority on dodging bullets, but there also exists the capacity for a ship that would try to seek bullets. There aren't often ships that persist beyond the initial 20 generations with this pathological behaviour.

Pickup collection works almost identically to bullet avoidance but in reverse, with closer pickups having higher priority over distant pickups. As it is also weighted by a weight that can be positive or negative, this too can result in greatly prioritising collection of pickups or can lead to pickup avoidance.

4.2 Executing Actions

Movement is based upon two stages. The first stage involves turning on all thrusters that would move the ship in the calculated desired direction of movement. On initialisation of the controller, the thrust vectors from every combination of thrusters is calculated and stored. After checking the thrust vector associated with every move action, the move action producing the thrust vector most matching the desired thrust is used. All thrusters associated with the move action are activated for the simulation step.

The second stage is only used if there is currently an enemy target. If there is, the second stage involves activating any additional thrusters that would help with spinning a turret in range of a target. All turrets are checked for their direction of fire and compared to the direction vector from the ship to the target. The smallest angular difference is used to pick the closest turret, and then all move actions are checked again for actions that would individually produce the best torque as desired for the angular difference. This is then used to fire any additional thrusters in addition to stage one to spin the ship.

Once movement is handled, all turrets are checked to see if firing would collide with a stationary hostile ship. If they would, the turret is fired.

5 Evolution

CMA-ES is used as the main evolutionary algorithm for improving spaceships, with an initial mean X of 0 and an initial standard deviation of 100. The standard deviation affects spatial dimensions the most apparently, and the majority of chromosome dimensions map directly to the physical space of the game world. CMA-ES is run for 10,000 evaluations before terminating.

5.1 Structure of Chromosome

The spaceship chromosome as it stands now is a 39-dimension array of doubles, each interpreted and converted outside of the main CMA loop. As individual dimensions relate to different elements that work better in different powers of ten, these doubles are scaled up and down as appropriate for the purpose of the represented value.

The first three dimensions indicate the initial position as a 2D co-ordinate and the initial rotation of the ship. These values are not scaled, and are used as the frame of reference for scaling the other dimensions. These positions are also given relative to the centre of the game world as opposed to a corner. The next four dimensions relate to controller weightings. Each of these values is scaled by a factor of 0.01, and themselves operate as weights on the controller's sub-objectives. After this are five sets of four values, a set of values for each of the five spaceship components. The first of these values is simply used to determine whether a component is a turret or a thruster, and the remaining three values determine an attachment position offset and rotation local to the spaceship's own position and rotation. These physical properties are scaled by a factor of 0.1 from the real values held in the chromosome.

5.2 Problems

The problem studied appears to be incredibly complex and highly prone to a lot of noise and related factors. As the individual performance of a spaceship tied to a chromosome is entirely dependent on the performance of spaceships related to other chromosomes, it is difficult to tell at all what the performance of a spaceship is. There exists no objective reward function that will indicate whether a given spaceship is good on an objective, non-relative scale. When confronted with this noise and shifting fitness landscape, the standard deviation of the population begins to grow and lead to very large, maladaptive spaceships. These large spaceships then perform poorly, but still are judged upon factors ultimately relative to all other spaceships, which also perform poorly. When a population begins to perform badly, it seems to be a repeating pattern that the entire population begins to rapidly diverge into solutions that do not seem effective on any objective scale.

5.3 Things for Consideration

It seems apparent, through writing this report, that one major thing that could be used to reduce noise within this simulation, and avoid the red queen issue of a shifting landscape of fitness, would be to introduce a totally new fitness measure that relies on some comparison to a static unchanging test case. This could be manifested in the sense of firing at static targets, or a series of fixed tests run sequentially for each ship, and may not even involve the combat simulation at all.

It may also be the case that CMA-ES is just not effective for this particular scenario, and that different evolution strategies might need to be considered.