Evolving Spaceship Designs for Optimal Control and the Emergence of Interesting Behaviour

Samuel A. Roberts, Simon M. Lucas

***Abstract*—This paper investigates the evolution of spaceship designs given three different fitness functions: one based on specified design objectives, and two simulation-based measures that assess how well a simple controller is able to steer the ships over a number of courses. We use a simple but well-grounded physics model which accounts for the mass of the components and their moments of inertia: the position of thrusters impacts on the linear and angular acceleration that the ship is capable of. We show that evolution is able to generate effective ship designs in this space and that the simple controller generates some interesting behaviours. The evolved ships could be used as a source of diverse alien ships in 2D arcade-style video games.**

I. Introduction

This paper describes a simple but novel approach to evolving 2D spaceship designs to be operated in a simple but well-grounded 2D physics simulation. Each design specifies the position and orientation of a number of fixed-power thrusters. We use evolution to find designs that aim to optimally balance linear acceleration, angular acceleration, and the directional range of linear accelerations (e.g. can a ship thrust sideways (or any other angle) as well as forwards). Our work lies at the simpler end of the artificial life spectrum, but working in this simple space has some important advantages. It means that the simulation code can be simple and efficient, enabling designs to be evolved rapidly. The physics model is very similar to the one used in 2D games such as Asteroids, so the evolved ships (together with their controllers) could be used directly within games of this type. The movement of the ships is believable, since it results from the combination of thrusters being on and off, which can easily be displayed.

A. Motivation

There are three distinct motivations behind this work: procedural content generation, artificial life, and the creation of novel game mechanics. We now look at each of these in more detail.

1) Procedural Content Generation: Content generation is perhaps one of the more costly areas of the video games industry. Content within the context of a video game is defined here to mean any part of a video game that is not the code, such as models for entities within the game world; the layout of levels the game requires, or sound and music. As the speed and performance of computer hardware improves, so does the technology video games operate on. As the technology improves, consumer expectations rise and must be met by a corresponding amount of content. These rising expectations on the amount and depth of content mean a rising cost to meet these expectations.

"Procedural content generation" is a broad description of a variety of methods for allowing this growing need for content to be partially or fully satisfied by delegating the work of content creation to a system.

Many approaches to the problem of generating content have been explored and analysed in various types of content: from the tracks in racing games [1], weapon designs in a space fighter game [2], [3], music [4] and the underlying mechanics of games themselves [5], [6].

Out of the many techniques of procedural content generation, a significant number can be categorised as *constructive* or *generate-and-test* [7]. *Constructive* methods of content generation are outside the scope of this paper. *Generate-and-test* methods generate content and evaluate its fitness for inclusion in the game, generating content until a satisfactory candidate solution is discovered. It is this approach that we investigate in this paper, by evolving spaceship designs that can best be controlled by a greedy heuristic controller.

2) A-Life and Novelty: It has previously been demonstrated that evolutionary algorithms can generate entities for a simulated world. The solutions produced by such means have often produced interesting and novel entities with characteristic behaviours, such as those discussed in the classic Sims paper [8] on dynamically evolved creatures. While these entities do not directly interact with each other, as is common to many artificial life systems, they must compete against each other in order to propagate through subsequent generations. The spaceship designs produced in the course of this research have also demonstrated interesting properties and adaptive behaviour distinct for a particular scenario. This makes the experiment interesting in its own right as well as a potential means to an end, as the interaction between generated designs and controller leads to some interesting and at times unexpected results. Sometimes spaceships can move in swaying arcs, or refuse to move at all. The relationship between the design and the controller could be viewed as a relationship similar to that between a fixed brain and a varying body, with the behaviour resulting as a consequence of this relationship.

3) Novel Game Mechanics: Currently we are only evolving the number, position, and orientation of a set of rocket motors (thrusters) on a spaceship. Due to the simple but accurate (given the limitations of what we choose to model) implementation of a physics model, this leads to some interesting behaviours emerging, which already could provide some interesting enemy ships for an Asteroids style game. Each thruster is either on or off for each time-step, and in our opinion this adds interest to the behaviour compared to ships that move using unseen and perhaps unrealistic forces. The possibility also arises for novel (or at least, rarely used) game mechanics where ships can be partially disabled in a physically realistic way by shooting out particular thrusters. For example, if a ship has a single thruster acting as a brake, then shooting out that thruster may cause more havoc behind enemy lines than destroying the entire ship.

For now we leave novel game mechanics as interesting possibilities. In this paper we explore the extent to which effective ship designs can be evolved given our experimental setup.

B. Prior Research

The scenario of spaceships moving through two dimensional space was inspired by prior research [9], [10], [11] involving similar generation and simulation of ship designs in a similar environment. However, that research did not explain in detail either the physics model or the control algorithm for steering the ship. We believe that both aspects are very interesting, since they greatly affect the types of ship that will evolve. We also compare results from static design objective optimisation with results from simulation. We use Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [12], to evolve the position, orientation and (via a simple "build" process) the number of thrusters on a ship.

II. Methodology

A. Simulation Physics

1) Physical Properties of a Spaceship: A spaceship has a number of physical properties. At its most basic, a spaceship is modelled as a point. This point possesses a mass, m, a centre of mass, $\mathbf{R}$, and a moment of inertia, I. These values begin identical for all spaceships at the constant core values of m_c and I_c. A spaceship has a position vector, $\mathbf{s}$, a velocity vector, $\mathbf{v}$, a rotation, θ, and an angular velocity, ω. A spaceship can also have an arbitrary number of components attached to it, which are also modelled as points with their own masses, m_t, and turning moments, I_t. A component is attached to the ship based on a relative position offset, $\mathbf{s_t}$, and a relative angle offset, θ_t. The values specified for these are assumed to be when the spaceship is not rotated and its local axes match the world axes. The component attachment is rigid and cannot be broken for any reason.

The attachment of a component to the spaceship increases the overall mass of the spaceship through the modelling of an added "strut" mass. A strut affects the mass and turning moment of the overall ship depending on how long it is. The length of a strut, l is exactly equal to the distance from the point mass of the component to the spaceship's origin point. This is not the same as the spaceship's centre of mass, which is recalculated for each new addition to the ship. As the position offset of the component is relative to the origin point, this therefore means that $l = |\mathbf{s}_t|$. The mass of a strut is proportional to its length, or $m_s = k_m l$, where k_m is set to two. As the strut is rectangular, its centre of mass, $\mathbf{R_s}$, is exactly equal to its midpoint, which is $\frac{1}{2}\mathbf{s}_t$. Its moment of inertia is equal to its mass multiplied by an arbitrary constant, k_I. It is not modelled as a physical rectangle, however, and is instead modelled as an additional point mass halfway between the origin of the ship and the position of the thruster. All of these changes are in addition to the direct changes caused by adding the component itself to the ship.

In summary, the properties of the spaceship are defined as follows. Given that, for the ith thruster:

$$m_s^i = k_m |\mathbf{s}_t^i|$$

$$\mathbf{s}_s^i = \frac{1}{2}\mathbf{s}_t^i$$

$$I_s^i = k_I m_s^i$$

The properties of a ship with N thrusters are:

$$m = m_c + m_t \sum_{i=1}^{N} m_s^i$$

$$\mathbf{R} = \frac{1}{m}\sum_{i=1}^{N}(m_t \mathbf{s}_t^i + m_s^i \mathbf{s}_s^i)$$

$$I = I_c + \sum_{i=1}^{N}((I_t|\mathbf{s}_t^i - \mathbf{R}|^2) + (I_s^i|\mathbf{s}_s^i - \mathbf{R}|^2))$$

The main components used in this paper are thrusters. All thrusters have the same mass, moment of inertia and thrusting power. Thrusters can either be on, and providing a fixed thrusting force, or off, providing no force. The front of each thruster is defined to be its attachment point, and the resultant direction of thrust is in line from the back to the front of the thruster. Depending on the attachment position and rotation relative to the spaceship's centre of mass, thrusters can provide linear acceleration, angular acceleration, or in most cases both.

The shape of the spaceship's hull is defined as a polygon, with each attached component defining a vertex of the polygon. The shape of the spaceship does not affect the physics of the simulation, however, as the spaceship is modelled as a connected group of point masses. The model includes a set of struts linking each thruster to the centre of mass of the ship, as explained later.

2) Physical Environment: The environment that spaceships are simulated within is a two dimensional flat plane. There are no limits or edges to this plane, other than the numerical limits that arise from hardware. There is a velocity loss factor similar to friction. This is calculated as a multiplication per simulation step of both linear and angular velocities by a constant factor. This applies to all spaceships. Though non-intuitive given the scenario of spaceships in an empty space, this friction presents an obstacle for spaceships to overcome or a feature to exploit. This friction can also be set to zero if desired.

B. Spaceship Representation

As the core mass of each spaceship is constant, the only variation between spaceships lies in the arrangement of components attached to a spaceship. As a result, all that must be modelled to evolve a spaceship are the components, and how they are arranged. As the only existing components are thrusters, this reduces the requirements of the representation to simply specify the properties for the thrusters.

Each thruster has two properties (three numbers in total), as thrust is fixed at a constant value for all thrusters. These properties are a vector representing the position of the thruster and a real number representing the rotation of the thruster. Both values are relative to the origin point of the spaceship the thruster is attached to, providing the spaceship is aligned with the world axes. The vector representing position contains two elements, an x co-ordinate and a y co-ordinate, and this means in total three real numbers can be used to position and rotate a thruster.

As a result, a spaceship is represented as a vector of real numbers of size $3n$, where n is the maximum number of thrusters a spaceship can have. This may not always be equal to the number of thrusters a spaceship can possess. When constructing a spaceship from its representation, if a thruster would be placed within a certain distance of another thruster, it is silently discarded and not added to the spaceship. Thrusters occurring earlier in the vector have higher precedence in placement than thrusters occurring later, and are more likely to be considered valid.

Due to invalid thrusters not being placed, part of the representation of a spaceship can become unused data that does not directly affect the spaceship's performance. Despite the data for these impossible thrusters seeming useless, it can, as in nature, become useful if the data is altered significantly. It can also become useful if the data for thrusters preventing its placement are also removed.

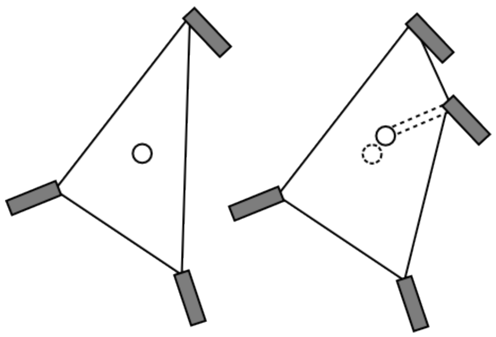


Fig. 1: When a component is added to a spaceship, the centre of mass, among other properties, changes. Further properties are affected by the addition of an invisible "strut" between the thruster and the centre of mass. This ensures larger ships have a larger mass.

C. Spaceship Controller

When being simulated, a spaceship must be controlled effectively in order to demonstrate that its design is useful for the simulation it exists in. The controller used for spaceships during simulation is based on a simple algorithm. It is not the same as a one-step look-ahead controller that would examine every possible state after every possible action a ship can take. Because this controller does not examine states, it is also considerably faster than a one-step look-ahead controller. This is because no simulation is required to determine states after actions have been made, which lowers the required amount of simulation in an experiment significantly.

During initialisation, a controller is assigned to each ship. Upon being assigned to a ship, a controller calculates a table of actions and the results that arise from taking these actions. As mentioned before, an action is a combination of states for the thrusters of a ship, as shown in Fig. 3. The controller is capable of deciding what thrusters should be active and inactive for each step of the simulation, and knowing in advance what each combination does allows for faster decisions later. For each action the ship can possibly take, the controller stores the resulting changes in linear and angular velocity.

For each step of a simulation, a controller is given a target position in the two dimensional space to move the ship towards. For simple linear movement, the target is the current position of the ship, plus an offset to the right. This unattainable target drives the ship constantly to the right, which is sufficient for the purposes of that particular experiment. For following a path, however, the target is whichever way-point the ship has to visit next along the path.

Given a target, the controller will first attempt to look up the action it can take that will give it the most appropriate magnitude of linear thrust towards the target. The most appropriate magnitude of the vector of acceleration is determined to be that of the magnitude of the distance. This is a rough estimate of the amount of thrust required and not a precise requirement, but this heuristic value aids the controller in selecting an appropriate amount of thrust to use to move to a target. During this selection process, only the magnitude is important, as the direction is considered in the next step. The action that produces a thrust vector with the closest (smallest absolute difference) magnitude is selected.

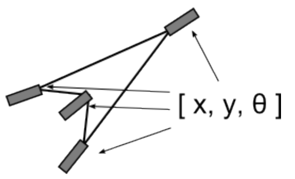


Fig. 2: Every thruster on a ship can be represented as a three values. The first two values relate to the position vector, and the third value relates to the rotation in radians. A string of sets of three values can represent a set of thrusters. As all other properties are constant, a set of thrusters can define a spaceship design.

Once it is known which action will produce the most appropriate magnitude of thrust, the direction heading of the target is compared to the direction this action will take the spaceship. If the angular difference between the direction to the target and the direction of thrust is within a certain tolerance, the controller will commit to accelerating the ship using the previously chosen thrust action.

If the angular difference is not within this tolerance, and the controller is not already committed to moving in a straight line, the controller will next consider the angular difference. It will look for an action with as suitable a torque force as possible, with a suitable torque being something close to the angular difference. The controller will then choose this action for the ship to take. If, while accelerating linearly, the angular difference becomes larger than some tolerance, the controller will cease accelerating linearly and attempt to correct its orientation.

The behaviour that results from this is perhaps not the best means of piloting a spaceship, but it is reliable and consistent. Spaceships will turn until they are in a suitable orientation to move rapidly towards a target, occasionally correcting their course.

It is worth noting that the controller has been known to struggle with certain ship designs (see Fig 4). The designs which tend to provoke this erroneous behaviour are not typically designs which would be useful or desired for the specific situation. However, they might also be designs that human players could control where the artificial controller could not. These are rarer occurrences, however.

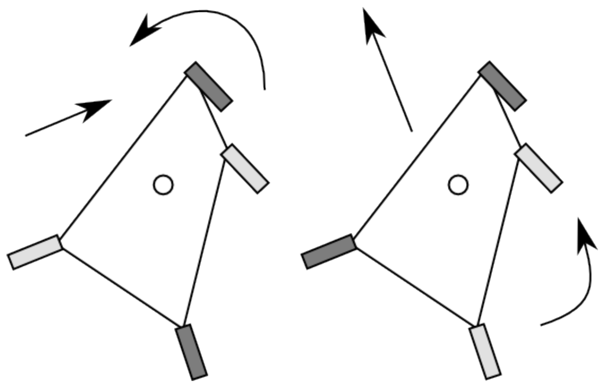


Fig. 3: In this diagram, lighter coloured thrusters are active. The curved arrows are representative of how much torque is produced and in which direction. The straight arrows are representative of how much thrust is produced and which direction the ship will move in.

D. Fitness Functions

Multiple fitness functions are used to gather different results, and to discover and compare the exact impact of specific requirements on the resulting spaceship designs. The first fitness function is an attempt to evolve a population to meet specific constraints likely to promote good ship design. Attempting to fulfil design objectives to produce ships that may be effective is, however, not identical to producing ships that are effective in practice, especially given the limitations of our simple controller. Two further fitness measures are used to produce ship designs, and both of these are empirical, relying on the outcome of fixed length simulations. As the implementation of CMA-ES minimises (rather than maximises), by default the fitness scores are converted from maximisation problems to minimisation by a simple transformation $F_t = C - F_a$, where F_t is the transformed fitness, F_a is the original fitness and C is a suitably large constant. On the graphs in the results section we show the raw fitness values.

1) Simple Design Objectives: The simplest of these fitness measures rewards spaceship designs with the capacity for high movement and high turning speed. These are calculated by examining every possible action a spaceship can take. An action is defined as a set of states for each of the thrusters belonging to a ship. As each thruster can be only on or off, the only means of controlling a spaceship is to set thrusters to be firing or not firing. For every possible action a ship can take, the resulting linear velocity and angular velocity are compared with the largest known linear and angular velocities, and the largest known values for each are updated respectively. Spaceships capable of higher linear acceleration and higher angular acceleration are judged to be more capable of moving and manoeuvring than ships without such properties, and are scored more highly. The final calculated fitness of a ship is:

$$F_a = w_1 v_{max} + w_2 \omega_{max}$$

where v_{max} is a scalar indicating the largest magnitude of linear thrust the ship is capable of, ω_{max} is a scalar indicating the largest absolute amount of angular acceleration the ship is capable of producing and both w_1 and w_2 are weightings for both of these components.

2) Linear Movement Experiment: A more general attempt to produce designs that may or may not fit an arbitrary scenario could likely be out-performed by evaluating a spaceship design through a simulated situation. As a result, an experiment and accompanying fitness measure is designed to produce ships capable of turning from an arbitrary starting rotation and moving from an arbitrary starting position as far to the right as possible. This promotes ship designs that are capable of quickly orienting themselves to a preferred direction of travel, as well as quick movement in that direction of travel, as in Fig.

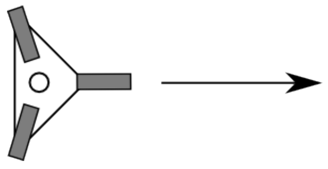


Fig. 4: This ship will never move to the right. None of its potential thrust vectors can be rotated to point in the desired direction, as all of its thrusters are directly aligned with the centre of mass. The ship cannot turn, and so this ship will never move.

5. Ships are scored based on the horizontal distance travelled within a fixed amount of simulation steps, and penalised for travelling vertically. This places emphasis on travel in a straight line. A high horizontal distance implies that the spaceship was able to turn and move quickly, and so is scored more highly. A high vertical distance implies the spaceship may not have altered its initial orientation correctly. The ship is simulated a number of times in specific rotations, and its final score is the mean of the scores achieved in all of these runs. For n trials, the initial starting angle for a ship on the ith repeat is $i\frac{2\pi}{n}$ radians. The fitness calculated during one of these trials is

$$F_i = w_x d_x - w_y |d_y|$$

with d_x as the horizontal distance travelled, d_y as the vertical distance travelled, and both w_x and w_y as weights. The final fitness is the mean of F_i over all trials.

The need for a series of specific starting rotations is to avoid random noise affecting the progress of the ship designs. Introducing random orientations can mean that poorly designed ships unable to steer correctly can be scored with a much greater fitness than they otherwise should have, while better ship designs could be scored far lower than they should be for similar reasons. Additionally, it is preferable to have a number of trials rather than a single fixed direction of rotation. If all ships were to be scored when pointing a fixed direction, it could produce designs that were only effective in moving in a single straight line. Although this experiment favours straight line speed, the fact that for some orientations it is necessary to turn first means that the fit designs need at least some turning ability.

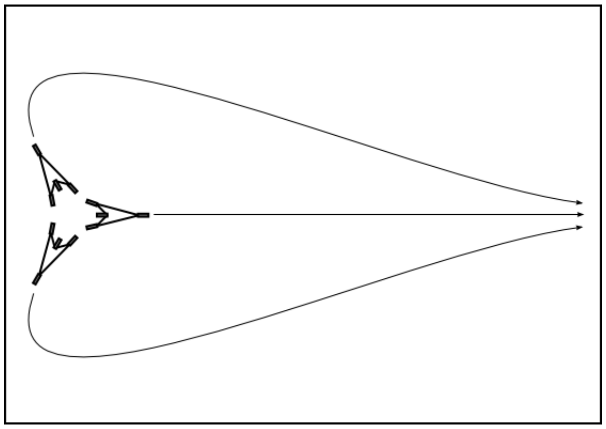


Fig. 5: Initial starting rotations of a ship in the linear movement experiment and expected trajectories.

3) Path Following Experiment: While assessing ships moving to the right does allow for spaceship designs that can turn and move, it does not promote designs that are capable of finer manoeuvring, such as turning to stay on a path. The path following experiment, along with fitness measure, exists to promote the development of ships capable of following a path consisting of a number of nodes or way-points, such as in Fig. 6. The path is kept constant throughout the course of the experiment, and the challenge posed to the ships is to reach as many way-points within a fixed amount of simulation steps. This path is created through generating a fixed number of points using a fixed seed for the random number generator used. Ships are scored based on the amount of way-points successfully reached, multiplied by a constant, and are then penalised by the distance to the immediate next way-point along the path that the ship did not reach. This allows for a finer degree of granularity in scoring ship designs than simple measurement of way-points reached alone. As with linear movement, the ship is simulated a number of times in different starting rotations, with its final score equal to the mean of the scores achieved over all trials. The starting rotation for each trial is, as with the linear movement experiment, $i\frac{2\pi}{n}$ radians, where n is the number of trials and i is the current trial. The fitness score for this trial is:

$$F_i = w_1 \cdot (N_{max} - N_j) + w_2 s$$

N_j is the index of the most recent way-point visited, N_{max} is the number of way-points on the path, and s is the magnitude of the displacement from the ship's ending position towards way-point N_{j+1}. w_1 and w_2 are weights used to adjust the significance of factors in the fitness measure. As with the linear movement experiment, the overall score is simply the mean of F_i over all trials. Unlike other fitness functions detailed in this paper, this particular function produces lower values for higher success. In this experiment, the aim is to minimise.

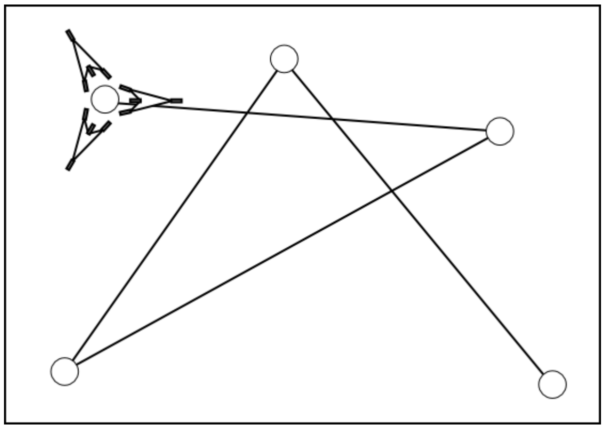


Fig. 6: Initial starting rotations of a ship in the path following experiment, and an example path that must be followed. Ships must move from one point to the next.

E. Software Implementation

All of the spaceship simulation was implemented for the purpose of this research in Java. We considered using a physics engine such as JBox2D[1], which is more than capable of running our simulations with sufficient accuracy. However, we chose to implement our own system for two reasons. Firstly, although we've not performed a direct speed comparison, we expect our simulation to be significantly faster due to its

[1]http://www.jbox2d.org

extreme simplicity compared to JBox2D. Secondly, developing our own bespoke software allowed more rapid prototyping of ideas. For the evolutionary algorithm we used the Java version of CMA-ES[2] without modification, and using the default parameter settings.

TABLE I: Parameters and Constants

Experiment	Symbol
Design Objective	d
Linear Movement	l
Path Following	p

Parameter	Experiment	Value
Maximum generations	all	10000
Initial values	all	0
Initial standard deviation	all	20
Velocity loss factor per step	all	0.99
Spaceship initial mass m_s	all	100
Spaceship initial moment of inertia I_s	all	1
No. of ship thrusters	all	5
Thruster mass m_i	all	12
Thruster moment I_i	all	0.5
Thruster force	all	200
Strut distance mass factor k_m	all	2
Strut distance moment factor k_I	all	0.1
Angular tolerance to start moving (controller)	all	$\frac{\pi}{32}$
Angular tolerance to stop moving (controller)	all	$\frac{\pi}{8}$
Weighting of maximum velocity w_1	d	1
Weighting of maximum torque w_2	d	1
Number of starting rotations n	l, p	3
Number of steps in a simulation	l, p	200
Weighting of horizontal distance travelled w_x	l	1
Weighting of vertical distance travelled w_y	l	0.5
Way-points reached weighting w_1	p	100
Distance remaining weighting w_2	p	1
Number of way-points on path N_{max}	p	6
Radius of way-point	p	20

III. Results

We now describe the results of evolving spaceship designs to meet the three alternative fitness objectives. In addition to these quantitative results, we have also posted a video which illustrates some example behaviours of the evolved ships[3]. As can be seen in all of these experiments, as in Fig. 7, 8 and 11, the best fitness in each generation becomes a limit of the population fitness as a whole quite rapidly. After beginning with a high level of fluctuation as various solutions are found, the general population begins to converge to a single design. When this occurs, the best fitness measure found so far is not as prone to change as much as the rest of the population does over time.

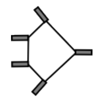
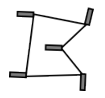
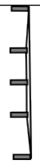
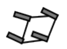
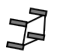
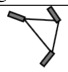
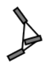
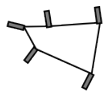
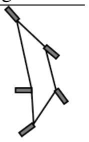
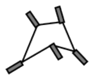
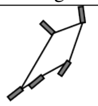
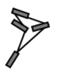
A. Simple Design Objectives

Spaceship designs produced in this context are almost always small. This is perhaps a result of smaller ships possessing smaller mass, and therefore greater acceleration potential. This greater acceleration potential results in spaceships which, when compared against similar configurations of thrusters for larger ships, accelerate at a higher rate than larger ships. It is interesting to note that the designs produced follow certain zig-zag patterns. Results of a typical evolutionary run are shown in Figure 7.

[2]http://www.lri.fr/~hansen/cmaes_inmatlab.html

[3]http://www.youtube.com/watch?v=3eAEbizMVPw

TABLE II: Example Ship Designs, with Fitness Scores

Fitness	Off-line Designs		
			
Design Objective	7.99	8.27	8.51
Linear Movement	1774	1866	2114
Path Following	738	785	864
Fitness	**Design Objective Designs**		
			
Design Objective	6.53	6.52	7.85
Linear Movement	1855	1590	1934
Path Following	786	845	911
Fitness	**Linear Movement Designs**		
			
Design Objective	7.47	8.59	8.75
Linear Movement	1508	1646	1737
Path Following	739	777	755
Fitness	**Path Following Designs**		
			
Design Objective	8.52	8.50	8.64
Linear Movement	1744	1795	1728
Path Following	575	671	638

B. Linear Movement Experiment

Spaceship designs that arise here tend to have multiple thrusters concentrated on a single side of the ship in order to provide a greater thrust in a single direction. The thruster positions will also allow for turning, in order for the ship to orient itself in the direction that will allow it to move the fastest. It is clear, though, that linear movement is prioritised, and the capacity to slow down quickly is often lacking. The spaceships produced in this context do not perform as well when tested in the path following experiment. A typical sample run is shown in Figure 8.

In addition to the standard linear movement experiment, Figure 9 demonstrates how adversely random starting orientations can affect the progress of a population. As can be seen, there is no improvement whatsoever, as the random factor has far outweighed every other aspect of the experiment, causing extremely unfit designs to be considered fit. This can lead to arbitrary designs, or it can lead to ships that cannot move or turn in a reasonable manner at all. Over time, this causes the fitness of the population as a whole to either reach zero, with

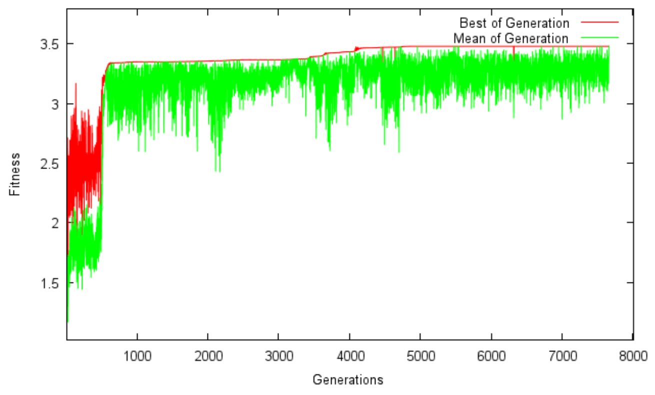


Fig. 7: An example of a swiftly converging design objective run. Due to lack of improvement near the end of the run, it was terminated early. It is common for design objective runs to converge to a maximum without advancing much further. The fitness shown is raw fitness that has not been transformed for CMA-ES.

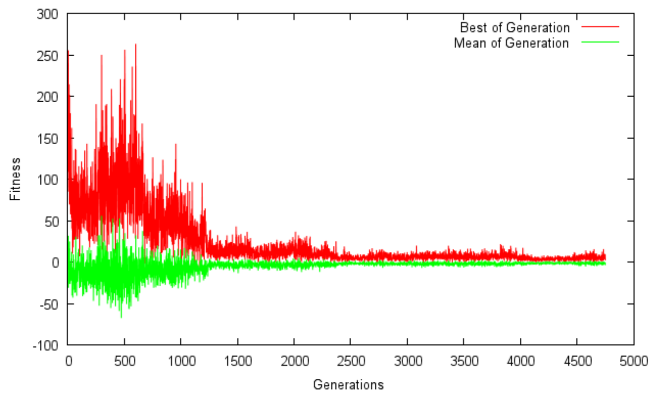


Fig. 9: An interrupted run of the linear movement experiment, with sub-run starting orientations set to random angles. As can be seen, the population does not improve in a consistent manner and the run has been cancelled due to inability to meaningfully improve the population.

minor fluctuations as further ships are erroneously considered fit, or to display no trend other than uniform random values.

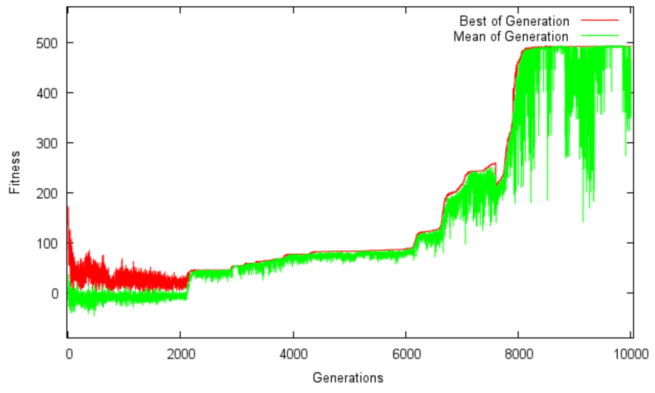


Fig. 8: The typical progress of a spaceship population attempting to move to the right, shown here over 10,000 generations. Overall population convergence to an effective design is achieved by generation 8000. The fitness shown here is raw fitness that has not been transformed for CMA-ES.

C. Path Following Experiment

The path following experiment is a slightly more difficult scenario to produce effective spaceship designs through evolutionary means. The produced ships must be able to move quickly along the straight lines between way-points on the path, as well as turn to the next way-point as quickly as can be managed. Selection pressure favours ship designs that are capable of moving and turning very fast, and ships that succeed in this particular scenario tend to be small, with thruster positions allowing for a higher turning moment over a prolonged acceleration in one direction.

Due to the nature of the fixed path, it is common to find ships that exploit the shape of the path. This is possible overfitting. In particular, given a path where a circular turn is the best course of action for the way-points closest to the start, ship designs emerged that proved adequate at turning a circle, but struggled to proceed onwards from the circular turn.

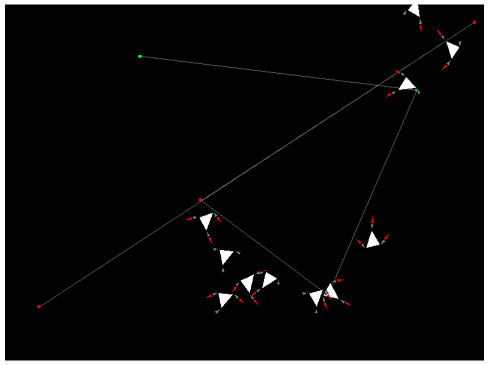


Fig. 10: Visualisation of a group of ships, 3000 generations into an example run, attempting to follow a path. Red nodes have not yet been visited by any ships, while green nodes have been visited by at least one ship. It is apparent here that the population has converged towards a specific design.

D. Ship Shapes

We observed an interesting feature of how ship designs evolved during both the linear movement and the path following experiment. During early generations the ships tend to be small. This is a consequence of the chosen standard deviation of the initial population. Over time, they evolve into large bloated ships that are able to orient themselves but move slowly. These are typically observable after 500 generations. Then, usually by 2000 generations smaller and fitter ships have evolved that are able to use all or most thrusters effectively but without the excessive mass due to the long struts needed for larger ships. They can be observed in the video previously referenced, and an example is also shown in Figure 12.

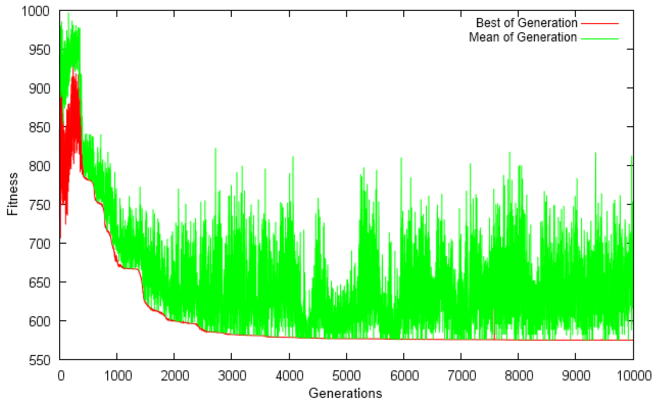


Fig. 11: Displayed here is an example of a path following run showing high convergence to an effective design. Little improvement is seen once the population becomes saturated with minor variations on a single effective design.

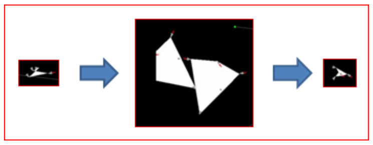


Fig. 12: Frequently observed evolution of ship shapes over time. In the early generations (left image) ships are small with poorly coordinated thrusters, or missing thrusters due to overlap. After around 500 generations ships are often bloated in size (two sample ships shown in middle image). Then after several thousand generations (right image shows a ships after 5000 generations) small ships are found that are able to use most or all thrusters in effective combinations.

IV. Conclusion

In this paper we have evolved spaceship designs in a simple but physically realistic manner. We used three fitness functions: a design objective one, and two simulation based ones. CMA-ES was found to be an effective algorithm for evolving spaceships in accordance with the specified fitness function, where each spaceship design was encoded as a vector of 15 numbers. However, it was necessary to use a fixed set of initial spaceship orientations, since random orientations meant that the signal to noise ratio was too low when measuring fitness. The resulting spaceships could be used directly within an Asteroids style video game, providing a rich variety of enemies or collaborators who can be shot at and partially disabled in interesting ways.

We also found the experimental design space to be interesting. The controller, the spaceship representation, the physics model and the fitness function all interact to provide the environment for evolution. In future we are interested in exploring the effects of the specific control algorithm on the evolution of the spaceship designs. However, the final designs in most cases are not aesthetically very pleasing (in our opinion), and usually don't appear to be as fast as they could be. For example, in the path following experiment many of the best solutions found do not have many thrusters on most of the time. We hypothesise that given a smarter controller, it may be possible to evolve more aesthetically pleasing spaceships within the same experimental setups; this would truly be a case where beauty comes from within.

Acknowledgements

Samuel Roberts is supported by an EPSRC PhD Studentship.

References

[1] D. Loiacono, L. Cardamone, and P. L. Lanzi, "Automatic Track Generation for High-End Racing Games Using Evolutionary Computation," *IEEE Transactions on Computational Intelligence and AI in Games, Vol. 3, No. 3*, 2011.
[2] E. Hastings, R. Guha, and K. Stanley, "Automatic Content Generation in the *Galactic Arms Race* Video Game," *IEEE Transactions on Computational Intelligence and AI in Games, Vol. 1, No. 4*, 2009.
[3] ——, "Evolving Content in the Galactic Arms Race Video Game," *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 2009.
[4] F. Pachet, "Beyond the Cybernetic Jam Fantasy: The Continuator," *IEEE Computer Graphics and Applications*, pp. 31 – 35, 2004.
[5] A. M. Smith and M. Mateas, "Variations Forever: Flexibly Generating Rulesets from a Sculptable Design Space of Mini-Games," *IEEE Conference on Computational Intelligence and Games*, 2010.
[6] J. Togelius and J. Schmidhuber, "An experiment in automatic game design," *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 2008.
[7] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based procedural content generation," *IEEE Transactions on Computational Intelligence and AI in Games*, 2011.
[8] K. Sims, "Evolving Virtual Creatures," *Proceedings of SIGGRAPH*, pp. 15 – 22, 1994.
[9] A. Liapis, G. N. Yannakakis, and J. Togelius, "Optimizing Visual Properties of Game Content Through Neuroevolution," *Seventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2011.
[10] ——, "Neuroevolutionary Constrained Optimization for Content Creation," *IEEE Conference on Computational Intelligence and Games*, 2011.
[11] ——, "Adapting Models of Visual Aesthetics for Personalized Content Creation," *IEEE Transactions on Computational Intelligence and AI in Games*, 2011.
[12] N. Hansen, "The CMA Evolution Strategy: A Comparing Review," *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms*, pp. 75 – 102, 2006.