Chapter 15
Evolutionary Algorithms towards Generating Entertaining Games

Zahid Halim and A. Raif Baig

Abstract. Computer games are gaining popularity by every passing day. This has increased the number of choices in computer games for the users. At the same time the quality of entertainment provided by these games has also decreased due to abundance of games in the market for personal computers. On the other hand the task of game development for the developers is becoming tiresome, which requires scripting the game, modeling its contents and other such activities. Still it cannot be known how much the developed game is entertaining for the end users. As entertainment is a subjective term. What might be entertaining for one user may not be entertaining for others. Another issue from the point of view of game developers is the constant need of writing new games, requiring investment both in terms of time and resources. In this work we create a set of metrics for measuring entertainment in computer games. The genres we address are board based games and predator/prey type of games. The metrics devised are based on different theories of entertainment specifically related to computer games, taken from literature. Further we use Evolutionary Algorithm (EA) to generate new and entertaining games using the proposed entertainment metrics as the fitness function. The EA starts with a randomly initialized set of population and using genetic operators (guided by the proposed entertainment metrics) we reach a final set of population that is optimized against entertainment. For the purpose of verifying the entertainment value of the evolved games with that of the human we conduct a human user survey and experiment using the controller learning ability.

1 Introduction

Nowadays computer games have became a major source of entertainment for all age groups, especially children. The reason for computer games being the primary source of entertainment could be many including these being highly interactive, high resolution graphics, diverse level of choice and challenge. According to

Zahid Halim · A. Raif Baig
National University of Computer and Emerging Science, Islamabad, Pakistan
e-mail: {zahid.halim,rauf.baig}@nu.edu.pk

N. Bessis, F. Xhafa (Eds.): Next Generation Data Technologies for CCI, SCI 352, pp. 383–413.
springerlink.com

a survey conducted in [1] on 1254 subjects, only 80 were found playing no electronic games in the last 6 months. The results in [1] show the popularity of computer and video games in young generation.

Thinking from the point of view of game developers it has always been a challenge to measure the entertainment value of the human player. This is due to the fact that entertainment is very subjective. It also depends upon the genre of game and contents of the game in addition to the subject (user) playing it. Keeping this fact in mind it would be very convenient for the game developers to develop entertaining games if they could somehow measure entertainment, the way other things like temperature, weight and many such things are measured. This would give a quantitative representation of the entertainment a game has as against the subjective one nowadays. Still based upon the measurable entertainment, the responsibility of producing a game like nowadays will be on the shoulders of game developer. Game developer will have to define the complete game from start till end along with each stage with its components and complexities. It would be very convenient that we could also produce the game automatically based upon the measurable entertainment. This would lead to interesting application in the area of computer game development.

1.1 Our Contribution

In this chapter we address the two issues in game development: (a) measuring entertainment value of a game and (b) automatic generation of entertaining games. We propose an entertainment metrics to quantitatively measure the entertainment value of the game. At present we address two genres of games and a separate set of entertainment metrics is proposed for each. The genres of games addressed include board based games and predator/prey games. There might be other sources of entertainment, other than the one we have considered for devising our entertainment metrics, like graphics and sound effects but these factors are not in the scope of the basic ingredients of a game and that is why they have not been considered. We have further shown the utility of the proposed entertainment metrics by generating new and entertaining games through computational intelligence techniques like genetic algorithms and evolutionary strategy which uses the proposed entertainment metrics as fitness function, guiding the evolution towards entertaining set of games. In order to counter check the entertainment value of the evolved games we have conducted a human user survey to verify that the results correlate with those produced by the system. In contrast to the user survey, the entertainment value of the evolved games is also verified using the Schmidhuber's theory of artificial curiosity [2].

Remaining of the chapter is organized as follows: section 2 covers background work, section 3 covers entertainment theories, section 4 lists search space, section 5 and 6 cover fitness function and chromosome encoding, respectively, section 7 explains the software agents, section 8 lists experiments and section 9 concludes the chapter.

2 Background Work

The concept of measuring entertainment and automatic generation of games and/or its contents is fresh and quite a limited amount of literature is available on the topic. This section is dedicated to the work done in the domain of measuring entertainment in computer games and their automatic generation. We have studied the work done in this regard by different researchers and listed it here.

2.1 Board Based Games

Iida [3], in 2003, has proposed a measure of entertainment for games and used it to analyze the evolution of game of chess over the centuries. This measure is considered to be the pioneer in quantification of entertainment. Even though Iida's work is limited to chess variants, the measure of entertainment can be easily applied to other board games. According to this measure, the entertainment value of a game is equal to the length of the game divided by the average number of moves considered by a player on his turn. The game is more entertaining if the value of this measure is low. The main idea is that the player should have many choices (moves) on the average and the length of the game should not be large. Long games with few choices per move are boring. The authors differentiate between possible moves and the moves considered by a player. The set of considered moves is smaller than the set of possible moves and the metric is based on the moves considered by a player.

In [4] the authors introduce the uncertainty of game outcome as a metric of entertainment. If the outcome is known at an early stage then there is not much interest in playing it. Similarly if it is found at the last move then it is probably probabilistic. The outcome should be unknown for a large duration of the game and should become known in the last few moves of the game. Authors state that it is easy to create new board games and variants of classical games but to make a game attractive to the human user, is challenging. In [4] a simple technique based on synchronism and stochastic elements is used to refine the game of Hex. Authors proof that the game's attraction has increased by conducting experiments to show an increment of the outcome uncertainty.

In [5] Symeon uses board games for e-learning. He proposes an e-learning board game that adopts the basic elements of a racing board game and cultivates in students skills like creativity, problem-solving, and imagination, as students try to reach the end by improving their performance in a variety of learning activities.

2.2 Video Games

Togelius [6] has presented an approach to evolve entertaining car racing tracks for a video game. Tracks were represented as b-splines and the fitness of a track depended on how an evolved neural network based controller (modeled after a player) performed on the track. The objectives were for the car to have made maximum progress in a limited number of time steps (high average speed), high

maximum speed (so that at least one section of the track is such that high speeds can be achieved), and high variability in performance (as measured by the final progression made) between trials (so that the track is challenging: neither too easy nor too hard). The game model used for experimentations in [6] is simple both graphically and physically (being 2D).

In [7] three metrics (which are combined into one) have been proposed for measuring the entertainment value of predator/prey games. The first metric is called appropriate level of challenge (T). It is calculated as the difference between the maximum of a player's lifetime and his average lifetime over N games. This metric has a higher value if the game is neither too hard nor too easy and the opponents are able to kill the player in some of the games but not always. The second metric is behaviour diversity metric (S). It is standard deviation of a player's lifetime over N games. It has a high value if there is diversity in opponent's behaviour. The third metric is spatial diversity metric E{Hn}. It is the average entropy of grid-cell visits by the opponents over N games. Its value is high if the opponents move all the time and cover the cells uniformly. This movement portrays aggressive opponent behaviour and gives an impression of intelligent game play. The three metrics are combined into one single metric $I = [\gamma T + \delta S + \varepsilon E\{Hn\}]/[\gamma + \delta + \varepsilon]$ where I is the interest value of the predator/prey game; γ, δ and ε are weight parameters. The work in [8] is some sort of extension of [7].

In [9], the authors have developed a computer game called "Glove" with three levels of incongruity: hard, easy and balanced. Their assumption is that the player would get frustrated or bored respectively, with the first two settings and would enjoy with the third one. The verification of this assumption has not been actually done in their paper. They argue that the actual complexity of a game can be defined as its difficulty level and the incongruity, i.e. the difference between the actual complexity and a player's mental complexity of a game can be measured indirectly by observing the player's behavior in the game.

In [10] an effort has been made to evolve rules of the game. The evolution of games in [10] is guided by a fitness function based on "learning ability". It gives low scores to games that do not require any skill to play and also to those which are hard and impossible whereas it assigns high fitness to games which can be learnt quickly. Although there are games being created automatically but they are not being measured against their entertainment value present in the game due to its rules and contents. They employ theory of artificial curiosity based fitness function introduced in [1] which focuses on the predictability of the game environment.

Chris in [11] modified the level generator to create new level on the basis of four parameters, three of which deals with the performing different operations with holes and the last parameter deals with the direction of the movement of Mario. If one or more direction switch is present, the level will suddenly be mirrored at random points, forcing the player to turn around and go the other way, until reaching the end of the level or the next direction switch. The aforementioned parameters are then categorized into two sub parts that are high or low thus making 16 possible combinations in total. In [11] several statistical features are noted during the playing of the game. These include completion time, time spent on

various tasks (e.g jumping), killed enemies (e.g way of killing) and information on how the player dies. Chris used neuroevolutionary preference learning of simple non linear perceptron to predict certain player emotions from game play features.

In [12] Nicola presents the concept of fun in the game of Pac-man based on the concept of flow. He argues that the fun factor in a game depends upon the psychological flow concept. Work in [12] deal with the question whether flow is a more reliable measure than asking human players directly for the fun experienced during the game. For the purpose of detecting flow a measure based on interaction time fraction between the human-controlled Pac-Man and the ghosts is introduced. The outcome of the measure is compared with the work done in this regard by Yannakakis and Hallam [13].

3 Theories on Entertainment

According to Csikszentmihalyi's theory of flow [14,15] the optimal experience for a person is when he is in a state of flow. In this state the person is fully concentrated on the task that he is performing and has a sense of full control. The state of flow can only be reached if the task is neither too easy nor too hard. In other words the task should pose the right amount of challenge.

In addition to the right amount of challenge, Malone [16] proposes two more factors that make games engaging: fantasy and curiosity. If a game has the capability of evoking the player's fantasy and makes him feel that he is somewhere else or doing something exotic then that game is more enjoyable than a game which does not do so. Curiosity refers to the game environment. The game environment should have the right amount of informational complexity: novel but not incomprehensible. Koster's theory of fun [17] states that the main source of enjoyment while playing a game is the act of mastering it. If a game is such that it is mastered easily and the player does not learn anything new while playing then the enjoyment value of that game is low.

Rauterberg [18, 19] has introduced the concept of incongruity as a measure of interest in a task. Given a task, humans make an internal mental model about its complexity. Incongruity refers to the difference between the actual complexity of the task and the mental model of that complexity. We have positive congruity if this difference is positive and negative congruity otherwise. In case of negative incongruity a person would be able to accomplish the task easily. Interest in a task is highest when the incongruity is neither too positive nor negative. In case of large positive incongruity the humans have a tendency to avoid the task and in situations of large negative incongruity they get bored. This requirement of right amount of incongruity is similar to the right amount of challenge in the concept of flow mentioned above. It has been further proposed that in case of reasonable positive incongruity the humans have a tendency to learn more about the task so that their mental model comes at par with the actual complexity of the task.

Several other related and derivative works are available on this topic. Many of them are covered in Yananakakis's recent survey [20].

4 Search Space

For the purpose of generating new games we need to define a search space that will be used by the evolutionary algorithm for this purpose, for these games to be entertaining the evolutionary algorithm will be guided by a fitness function, which will be our proposed entertainment metrics. As we are addressing two different genres of games we need to have separate search space and fitness functions for both.

4.1 Board Based Games

For defining the search space for board based games we use the search space of the popular board games of chess and checkers as a super set. Figure 1 summarizes the search space.

Search Space Dimension	Values
Play Area	Both white & black squares are used
Types of Pieces	6
Number of pieces/type	variable but at maximum 24
Initial position	First 3 rows & Both white & black
Movement direction	All directions, straight forward, straight forward and backward, L shaped, diagonal forward
Step Size	One Step, Multiple Steps
Capturing Logic	Step over, step into
Game ending logic	No moves, no king
Conversion Logic	Depends upon rules of the game
Mandatory killed	Depends upon rules of the game
Turn passing allowed	No

Fig. 1 Board based game search space dimensions.

The size of play area in our search space is a grid of 8x8 squares, alternating white and black, all squares can be used. Combining the rule space of chess and checkers we have total six types of pieces in our search space. Each type can have a minimum of 0 and a maximum of 16 pieces. However, total pieces should not be zero nor exceed 16. The initial positions of the pieces are the nearest three rows of a player. A cell can have one piece of type 0 to 6, type 0 means no piece is present in that cell. The search space to evolve new games consists of only those six movement logics as in both chess and checkers. Which include diagonal forward, diagonal forward and backward, movement in all directions, L shaped movement, straight forward and straight forward and backward. The step size for a piece can either be one or up to an occupied cell.

Capturing is done by jumping over or moving into the opponent's cell. The result of capturing is death of captured piece. The game ends if there are no more pieces left of a specific type. We call this type of piece the "piece of honour".

There can be zero or one piece type declared to be a piece of honour. A game ends if a piece of honour of any player is dead or the player without moves is the loser. A game can have a maximum of 100 moves. A piece may or may not convert to another type after reaching last row. Evolution decides which type is convertible. Each piece has a conversion logic which decides which type it will convert to when last row is reached. Turn passing is not allowed.

4.2 Predator/Prey Games

The predator/prey genre consists of one or more predators, predators may be homogeneous or heterogeneous in their behaviour, obstacles (which may or may not be for both predator and prey) and some objective for the prey to achieve. Pac-Man is a very popular game of this genre. Keeping the above constrained in mind and inspired by its closeness to the rule space defined by J. Togelius in his work [10]. We have defined our rule space as follows.

Play area consists of 20 X 20 cells. There are *N* predators of type *M*; each type is represented by a different colour. We have selected *N* to be 0-20 and *M* in range 0-3. Where colours are red, green and blue. Each type of predator moves around the play area according to any of the following three schemes: still, turn clockwise upon encountering an obstacle and turn counter clockwise upon encountering an obstacle. The predators may collide with each other and the prey. As the different predators may have different behaviour so the response to collision needs to be different of each type. The possible types of responses to a collision are as: death of the prey and/or predator, random change in current location of the prey and/or predator, no effect on the prey and/or predator. The score is calculated for the prey only, which is one of +1, 0, or -1 upon collision with a predator. For the predators for which upon collision with the prey, prey's score increases prey is predator for them and the predators are prey. This shows the uncertain nature of the game which adds some level of entertainment in the game as well. The time for which the game will be played vary from 1-100 time steps and the maximum score that a prey can achieve vary from 1-2000. The game will stop if any of the following is true: the time exceeds its maximum limit, prey has died, or the prey score exceeds the maximum score. Figure 2 displays a typical environment of one such game created based on the rule space defined. The yellow is the prey and remaining are the predators of different types.

Fig. 2 The play area of the predator-prey game.

5 Structure of the Chromosome

For the purpose of evolving games we have used Evolutionary Algorithms. Each individual chromosome of the EA population represents one complete set of rules for the game; whereas each gene of the chromosome represents one rule of the game.

5.1 Board Based Games

Based upon the above search space, the structure of the chromosome used is listed in figure 3. The chromosome consists of a total 50 genes. First 24 genes may contain values from 0 to 6 were 1 represents a piece of type 1, 2 for piece of type 2 and so on. Zero is interpreted as no piece. The piece type represented by gene 1 is placed in the cell 1 of the game board; piece type represented by gene 2 is placed in the cell 2 of the game board and so on.

Gene	Title	Value
1-24	Placement of gene of each type	0-6
25-30	Movement logic of each type	1-6
31-36	Step Size	0/1
37-42	Capturing logic move into cell or jump over 0/1	0/1
43	Piece of honor	0-6
44-49	Conversion Logic 0-6	0-6
50	Mandatory to capture or not	0/1

Fig. 3 Structure of the chromosome

Gene 25 to 30 represents movement logic for each piece type respectively, where 1 is for diagonal forward, 2 for diagonal forward and backward, 3 for all directions, 4 for L shaped movement, 5 for straight forward and backward and 6 for straight forward. Genes 31 to 36 are used for step size of each type, where 0 is used to indicate single step size and 1 for multiple step sizes. Genes 37 to 42 are used for step size of each type, where 0 is used to indicate step into and 1 for step over. Gene number 43 indicates the type of piece that will be the piece of honour, possible values include 0-6, where 1-6 indicate the piece type and 0 represents that there is no piece of honour in the game. Genes 44-49 represents the conversion logic, of piece type 1 to 6 respectively, when they reach the last row of the game board. Where 0 represents the piece will not be converted to any type and 1-6 represents the type of piece. The last gene represent whether it is mandatory in the game to capture the opponent piece in case it could be, 0 represents no and 1 represents yes.

5.2 Predator/Prey Games

There are a total of 30 genes in a chromosome; the chromosome encoding is shown in Figure 4. The rules of the game they represent and their possible values are as follows:

- First three genes each for representing the number of red, green and blue type of pieces. The values they can have range form 0-20.
- Next three genes each representing the movement logic of red, green and blue type of pieces. The possible values for these genes are 0 to 4 representing: still, clockwise, counter clockwise, random short and random long movement respectively.
- A total of next 15 genes for representing collision logic between two pieces or between any piece and an agent. Since there are a total of 4 entities (three pieces and an agent), hence the possible effects of collision between any two entities can be represented by a 4x4 -1 matrix. A collision between an agent and another agent is not possible because there is only one agent. Hence one element of the 4x4 matrix is empty and needs not be represented as a gene in the chromosome. The possible values of the collision logic genes are 0, 1 and 2 representing no effect on the colliding entity, the entity dies and are removed from the game and the piece is moved to some randomly chosen location respectively.
- Next 9 genes for representing the score addition or depletion on the collision of any two entities. The possible values of these genes are -1, 0, and 1.
- Next nine genes represent score effect for collision between: agent and red piece, agent and green piece, agent and blue piece, red and red piece, red and green piece, red and blue piece, green and green piece, green and blue piece, and blue and blue piece.

	Red	0-20		Blue-Green	0-2
	Green	0-20		Blue-Blue	0-2
Number of predators	Blue	0-20		Blue-Agent	0-2
	Red	0-4		Agent-Red	0-2
	Green	0-4		Agent-Green	0-2
Movement logic	Blue	0-4	Collision logic	Agent-Blue	0-2
	Red- Red	0-2		Red- Red	-1,0,+1
	Red- Green	0-2		Green-Green	-1,0,+1
	Red-Blue	0-2		Blue-Blue	-1,0,+1
	Red- Agent	0-2		Agent-Red	-1,0,+1
	Green-Red	0-2		Agent Green	-1,0,+1
	Green-Green	0-2		Agent-Blue	-1,0,+1
	Green-Blue	0-2		Green-Red	-1,0,+1
	Green-Agent	0-2		Blue-Red	-1,0,+1
Collision logic	Blue-Red	0-2	Score logic	Blue-Green	-1,0,+1

Fig. 4 Chromosome encoding along with possible values a gene can have.

6 Fitness Function

Each chromosome encodes the rules of a game. In other words, it is a complete game. The aim of the evolutionary process is to evolve a population of games and find a best one which is entertaining for the player. For this purpose we have assumed that better entertainment is based on four different aspects described below for each of the genre discussed. In three of these aspects (for board based games) we assume that both the players play each game with the same strategy (random controller). Hence both have the same chances of winning.

6.1 Board Based Games

6.1.1 Duration of the Game

In general, a game should not be too short or too long, as both are uninteresting. For example, if a game is such that it usually ends after a few moves (like Tic-Tac-Toe) then it would not appeal to adults. On the other hand, if a game usually continues for several hundred moves then the players may choose not to play it due to lack of enough time.

The duration of play (D) of a game is calculated by playing the game n times and taking the average number of moves over these n games. For the games evolved in this chapter, the maximum moves are fixed at 100 (50 for each player). If a game does not end in 100 moves then it is declared a draw. The average value of D is taken because if the game is played multiple times with a different strategy (or even by the same strategy which has probabilistic components) then we do not get the same value of D every time. For averaging, the game is played n = 20 times in our experiments. Equation (1) shows the mathematical representation of D.

$$D = \frac{\sum_{K=0}^{n} L_K}{n} \tag{1}$$

Where, L_K is the life of the game playing agent in game K. In order to reward games neither too short nor too long raw value of D is scaled in range 0-1. The boundaries for scaled value of D are shown in figure 5.

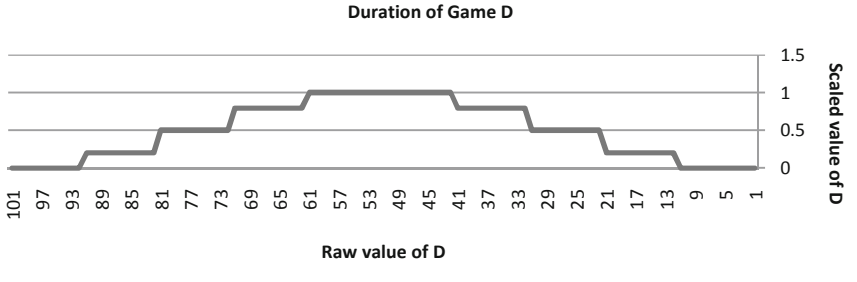

Fig. 5 Scaling ranges for raw value for duration of game.

For the raw duration of games 0 to 10 and 100 to 90 a scaled value of 0 is assigned, for ranges 11-20 and 81-90 a value of 0.2 is assigned, for 21-30 and 71-80 a scaled value of 0.5 is used, 31-40 and 61-70 are converted to 0.8 and a range of 41-60 is assigned the highest value i.e. 1.

6.1.2 Intelligence for Playing the Game

A game is interesting if the rules of the game are such that the player having more intelligence should be able to win. The intelligence (I) is defined as the number of wins of an intelligent controller over a controller making random (but legal) moves. For this purpose the game is played *n* times (*n* = 20 times in our experiments). Higher number of wins against the random controller means that the game requires intelligence to be played and does not have too many frustrating dead ends. Intelligence I is calculated using equation (2).

$$I = \frac{\sum_{k=0}^{n} I_K}{n} \tag{2}$$

Where, I_K is 1 if intelligent controller wins the game otherwise its 0.

6.1.3 Dynamism Exhibited by the Pieces

This aspect assumes that a game whose rules encourage greater dynamism of movement in its pieces would be more entertaining than a game in which many pieces remain stuck in their cells for the entire duration of the game. The dynamism is captured by the following fitness function given in equation (3).

$$Dyn = \frac{\sum_{i=1}^{n} \left(\frac{\sum_{i=1}^{m} (C_i)/Li}{m} \right)}{n} \tag{3}$$

Where,
$C_{i,}$ is the Number of cell changes made by piece i during a game
$L_{i,}$ is life of the piece i
And m is the total number of pieces specified in a chromosome.

The dynamism is averaged by calculating it for 20 games for the same chromosome. This fitness function has a higher value if the pieces show a more dynamic behaviour.

6.1.4 Usability of the Play Area

It is interesting to have the play area maximally utilized during the game. If most of the moving pieces remain in a certain region of the play area then the resulting game may seem strange. The usability is captured using equation (4).

$$U = \frac{\sum_{i=1}^{n} \left(\frac{\sum_{k=0}^{m} (C_k)}{|C_u|} \right)}{n} \tag{4}$$

Where,
C_k, is usability counter value for a cell k.
$|C_u|$, is the total number of usable cells.
n, is 20 as explained previously.

A usability counter is set up for each cell which increments when a piece arrives in the cell. The usability U is averaged by playing twenty different games for a chromosome. A cell which is never visited during a game will have a counter value of zero, thus contributing nothing to the usability formula. Furthermore, a cell which has a few visits would contribute less than a cell having large number of visits.

6.1.5 Combined Fitness Function

The above four metrics are combined in the following manner. All chromosomes in a population are evaluated separately according to each of the four fitness functions. Then the population is sorted on "duration of game" and a rank based fitness is assigned to each chromosome. The best chromosome of the sorted population is assigned the highest fitness (in our case it is 20 because we have 10 parents and 10 offsprings), the second best chromosome is assigned the second best fitness (in our case 19), and so on. The population is again sorted on the basis of "intelligence" and a rank based fitness is assigned to each chromosome. Similarly, rank based fitness is assigned after sorting on "diversity" and "usability". The four rank based fitness values obtained for each chromosome are multiplied by corresponding weights and then added to get its final fitness.

$$FF = aD + bI + cDyn + dU \tag{5}$$

Where, a, b, c, and d are constants. In our experiments we keep the values of these constants fixed at 1. The multiplication with a corresponding weight allows us to control the relative influence of an aspect. The calculation of rank based fitness gets rid of the problem of one factor having higher possible values than another factor.

6.2 Predator/Prey Games

6.2.1 Duration of the Game

The fitness function should be such that it discourages such a possibility. The duration of play, D, is calculated as in equation 6.

$$D = \frac{\sum_{k=0}^{n} L_K}{n} \tag{6}$$

Where, L_K, is the life of the game playing agent in game K ,

And n is the total number of times the agent plays a game represented by a single chromosome, in this case n is fixed to 20.

Since there are many probabilistic factors in the game, we will not get the same value of D if the game is played multiple times. For this purpose the game is played n times (n= 20 in our experiments) for a chromosome and an average is taken.

6.2.2 Appropriate Level of Challenge

The level of challenge of the game can be directly measured from the score of the player in the game. Higher a player score more is his interest and motivation to play the game again. Too high a score, achieved easily is not challenging enough and similarly too low a score even after an intelligent game play is discouraging. There has to be an appropriate level of challenge provided by the rules. The factor of uncertainty in the rules of the game where entities other than the agent can have positive and/or negative effect on the player's score, introduces a factor of good or bad luck in the game, as in snake and ladders game, and can enhance the enjoyment level, provided that this uncertainty factor is not too high. The challenge C is converted into a fitness function using equation 6:

$$c = e^{\left(\frac{-|S_m - S_a|}{S_m}\right)} \tag{6}$$

Where

$$S_a = \frac{\sum_{k=0}^{n} S_K}{n}$$

S_K, is score of the agent when it plays it K^{th} time.
n, is 20 as explained previously.

Since the value of S_a can also be negative, hence we use the following processing:

$$S_a = \begin{cases} S_a, & \text{if } S_a \geq 0 \\ |S_a| + 20, & \text{otherwise.} \end{cases} \tag{7}$$

6.2.3 Diversity

The diversity of the game is based upon the diversity of the pieces in the game. The behavior of the moving pieces of the game should be sufficiently diverse so that it cannot be easily predicted. Pieces with complex movement logic including random reallocation (teleport) would be more entertaining than static or simply moving ones. The diversity is captured by equation 8.

$$Div = \frac{\sum_{i=1}^{n} \left(\sum_{k=0}^{m} (\partial_k)\right)}{n} \tag{8}$$

Where,
m, is the total number of pieces (all three types) specified in a chromosome.
∂_k, Number of cell changes made by piece k during a game.
n, is 20 as explained previously.

6.2.4 Usability

Usability is the fourth and last factor we have considered for our metrics of entertainment. It is interesting to have the play area maximally utilized during the game. If most of the moving pieces remain in a certain region of the play area then the resulting game may seem strange. The usability is captured by equation 9:

$$U = \frac{\sum_{i=1}^{n} \left(\frac{\sum_{k=0}^{m} (C_k)}{|C_u|} \right)}{n} \tag{9}$$

Where,
C_k, is usability counter value for a cell k.
$|C_u|$, is the total number of usable cells.
n, is 20 as explained previously.

A usability counter is set up for each cell which increments when a piece arrives in the cell. The usability U is averaged by playing ten different games for a chromosome. The total cells for our current experimentation are 14x14 minus the 7+7=14 cells used by two walls.

6.2.5 Combined Fitness Function

To assign a chromosome a single fitness value we use rank based fitness using the above mentioned four metrics, as explained in section 6.1.5. The four rank based fitness values obtained for each chromosome are multiplied by corresponding weights and then added to get its final fitness, as in equation 10.

$$FF = \alpha_D + \beta_C + \chi_{Div} + \delta_U \tag{10}$$

Where, $\alpha, \beta, \chi,$ and δ are constants. In our experiments we keep the value of these constants fixed at 1. The multiplication with a corresponding weight allows us to control the relative influence of an aspect.

7 Software Agents

Evolutionary algorithm evolves a population of games and the fitness of each game has to be determined in each generation we may have a total of several thousands of such fitness evaluations. Since a fitness evaluation means playing the game several times, it is not possible to do so manually. We need software game playing agents. The more intelligent the agent the better will be the accuracy of fitness evaluation. We have developed two types of such agents for board based games (as required by the fitness function).

- Random agent.
- Agent using Min-Max with rule based evaluation function.

7.1 Random Agent

As the name suggests the random game playing agent plays the game by randomly selecting a legal move at each step. The agent follows the following algorithm listed in figure 6:

```
Input: Game Board current state
1.    Generate all legal moves
2.    Store the moves in a queue
3.    Shuffle the queue
4.    If Not mandatory to kill
5.        Randomly select a move from the queue.
6.    Else
7.        Select a move that captures an opponent's piece, if such move exists
8.        Otherwise, randomly select a move from the queue.
Output: Next move to take
```

Fig. 6 Algorithm for the random playing agent

The agent initially generates all the legal moves and stores them in a queue. The queue is shuffled once all the moves are saved in it. Shuffling is important, as we take an average of 20 games to calculate the individual metrics values, if the queue is not shuffled then each time the game is played it will use the same sequence of moves to play the game and fitness values will remain the same in each iteration of the game play. If the mandatory to capture bit is "on" in a chromosome which is being evaluated then the agent first tries to find a move that will capture an opponent's piece. If no such move is found it randomly selects a move from the queue.

7.2 Agent Using Min-Max with Rule Based Evaluation Function

This type of agent is intelligent as compared to the random one. It generates all the possible one ply depth game boards using a min-max algorithm. Each of the resulting game board is evaluated using a rule based evaluation function and the one with the highest evaluation is selected as a next move.

Evaluation function for this type of agent assigns priorities (weights) to piece-type according to whether its disappearance would cause the game to end, flexibility of movement (more directions and multiple step sizes are better), and capturing logic (capturing by moving into opponent's cell is better). Once the priority of a piece is calculated we multiply each piece with its corresponding weight and calculate weighted summation for self and opponent. The board evaluation is the self weighted summation minus opponents weighted summation. Figure 7 lists the algorithm for the evaluation function we use.

```
Input: Game Board current state
1.    For each piece
2.        priority=0
3.    For each piece
4.        if is piece of honor
5.            priority = priority +1 000
6.        if movement logic all directions
7.            priority = priority + 8
8.        if movement logic diagonal Forward and Backward
9.            priority = priority + 7
10.       if movement logic Straight Forward and Backward
11.           priority = priority + 7
12.       if movement logic diagonal Forward
13.           priority = priority + 6
14.       if movement logic Straight Forward
15.           priority = priority + 6
16.       if movement logic L shaped
17.           priority = priority + 5
18.       if capturing logic step into
19.           priority = priority + 4
20.       if capturing logic step over
21.           priority = priority + 3
22.   Count the number of pieces of Player A
23.   Multiply the number of pieces of a type with its relevant priority
24.   Count the number of pieces of Player B
25.   Multiply the number of pieces of a type with its relevant priority
26.   Calculate boardValue = WeightSumofA-WeightSumofB
27.   Check if the Piece of Honour is dead add -1000 to boardValue
28.   Check if the Piece of Honour is NOT dead add +1000 to boardValue
Output: boardValue
```

Fig. 7 Algorithm for evaluation of board positions

Since we are using mini-max of single ply hence we had to incorporate a mechanism in the evaluation function to overcome the randomness effect near the end of the game when pieces are few and may be far apart. In such cases the evaluation function listed in figure 7 gives same evaluation for all the board positions thus increasing the duration of game. To avoid such situation we restrict the agent to select the move which decreases distance between its own piece and one of an opponent's pieces, provided all next game board position have equal evaluations.

7.3 Agents for Predator/Prey Games

For predator/prey games the controller is implemented using an (Artificial Neural Network) ANN and a rule based controller, used to evolve two separate populations to study the effect of controller on evolution. For ANN based controller we have used a multi-layer fully feed forward connected neural network, architecture is shown in Figure 8. There are a total of 6 neurons in the input layer, 5 neurons in the hidden layer and 4 output layer neurons. Sigmoid activation function is used at each neuron. The weights on the edges range between -5 to +5. The input vector to the neural network is (Δxr, Δyr, Δxg, Δyg, Δxb, Δyb) where Δxr, Δyr, Δxg, Δyg, Δxb and Δyb represent agent's distance from nearest red type predator in x-coordinate, agent's distance from nearest red type predator in y-coordinate, agent's distance from nearest green type predator in x-coordinate, agent's distance

from nearest green type predator in y-coordinate, agent's distance from nearest blue type predator in x-coordinate and agent's distance from nearest blue type predator in y-coordinate. The ANN outputs a 4 dimensional vector (Nu,Nd,Nl,Nr) where Nu,Nd,Nl and Nr represents agent's next position for up, agent's next position for down, agent's next position for left and agent's next position for right. Agent moves in the direction having the maximum value.

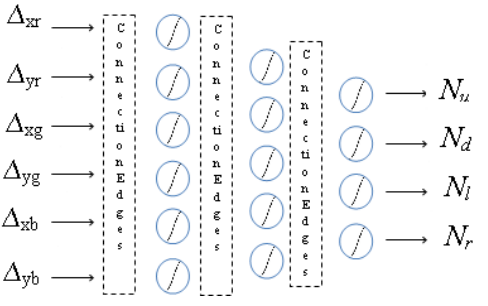

Fig. 8 ANN architecture used to control agent

For the purpose of training of ANN weights we have employed GA where each chromosome of the population represents the set of weights for the entire ANN. In this case the chromosome length is of 97 genes. Mutation is used only as a genetic operator. For each game a random population of GA representing the weights of the edges is created. The game is played 10 times using these weights and a score is assigned which is the average score achieved by the controller. The GA is run for 10 iterations and the best chromosome of each is saved. As the GA finishes we select the best chromosome out of 10, based upon the highest average score achieved.

The rule based controller is implemented as a human supplied rule set. The same controller is used for playing all games (chromosomes) during the entire evolutionary process. Our rule based agent controller is composed of rules formulated to implement the following policy. According to the game rules, at each simulation step the agent must take exactly one step. The agent looks up, down, left and right. It notes the nearest piece (if any) in each of the four directions, and then it simply moves one step towards the nearest score increasing piece. If there are no score increasing piece present it determines its step according to the following priority list:

- Move in the direction which is completely empty (there is only the wall at the end). If more than one directions are empty move towards the farthest wall (in the hope that subsequent position changes would show it a score increasing piece)
- Move in the direction which contains a score neutral piece. The farthest, the better.

- Move in the direction which contains a score decreasing piece. The farthest, the better.
- Move in the direction which contains a death causing piece. The farthest, the better.

Going into walls is not allowed, and if there is a wall present in the adjoining cell, the possibility of going in that direction is automatically curtailed. The above mentioned controller rules encourage the agent to maximize its score by trying to collide with the piece which increase its score and at the same time try to avoid collision with the rest.

8 Experiments

Different experiments carried out are explained in the sub-sections below for each of the game type.

8.1 Board Based Games

The methodology of the experimentations is such as we evolve new board based games using 1+1 Evolutionary Strategy (ES). Once we get a set of evolved games we first select minimum number of games, using equation (12), for analysis. In order to analyze and study the entertainment value contained in the evolved games we follow a twofold strategy i.e. by conducting a controller learnability experiment and a human user survey.

8.1.1 Evolution of New Games

In order to generate new and entertaining board based games we use 1+1 Evolutionary Strategy (ES). Initially a population of 10 chromosomes is randomly initialized with permissible values. The evolutionary algorithm is run for 100 iterations. Mutation is the only genetic operator used with a mutation probability of 30 percent. In each iteration of the ES one parent produce one child and a fitness difference is calculated between them. If it is greater than 4 (i.e. the child is at least half times better than its parent) child is promoted to the next population. We use the formula given in equation (11) to calculate the fitness difference.

$$Fitness\ Difference = \sum_{for\ all\ metrics} \left(1 - \frac{fitness_p - fitness_c}{fitness_p}\right) \tag{11}$$

Where,

$fitness_p$ is the fitness value of parent for current metrics
$fitness_c$ is the fitness value of child for current metrics

We keep an archive of 8 slots and in each iteration update it with the best 2 chromosomes based on each of the fitness metrics. Figure 9 shows the metrics values of one family of chromosome in shape of a graph (figure 9), over a period of 100 iterations.

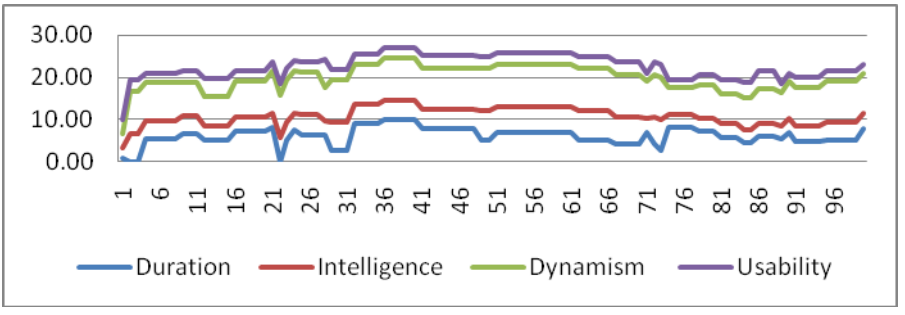

Fig. 9 Metrics values of a typical family of chromosome.

As we use 1+1 ES the best chromosome found in an iteration may get lost in iterations to come. For this purpose as mentioned previously we keep an archive of 8 for best 2 chromosomes against each of the metrics. As the evolutionary process progresses the number of changes, child beating its parent using fitness difference formula (equation (11)) decreases.

8.1.2 Games for Analysis

The evolutionary process gives 8 games evolved against the entertainment based on duration, intelligence, dynamism and usability. For further analysis of these games we select the set of most diverse games. Diversity (from each other) of these games is calculated using their fitness values and is listed in table 1 for one experiment.

Table 1 Fitness values of chromosomes in archive

		Game No.	Duration	Dynamism	Intelligence	Usability
Archive	Duration 1	1	0.885	0.081	1.000	21.051
	Duration 2	2	0.850	0.068	0.700	16.775
	Dynamism 1	3	0.021	0.181	1.000	22.065
	Dynamism 2	4	0.221	0.172	1.000	25.866
	Intelligence 1	5	0	0.086	1.000	23.085
	Intelligence 2	6	0	0.068	1.000	21.028
	Usability 1	7	0.400	0.066	0.850	84.927
	Usability 2	8	0.216	0.039	0.700	80.997

We calculate the diversity based on each of the four metrics for all pair of games using equation (12).

$$Game\ diversity = \left| \frac{Game\ Column\ Fitness - Game\ Row\ Fitness}{Selected\ Metrics\ Maximum\ Value} \right| \quad (12)$$

Selecting a threshold value of 0.6 table 2 shows the diversity count of evolved games. Diversity count indicates that a game is different from how many other games based on all the four metrics of entertainment.

Table 2 Diversity count of evolved games

Game No.	Different from number of games (for threshold ≥ 0.6)
1	5
2	5
3	3
4	1
5	0
6	1
7	6
8	3

Based upon the above statistics game number 1, 2 and 7 seems to be the most diverse. We select these three for further analysis. From this point onwards we will refer to these as game 1, 2 and 3 respectively. Rules of these games are listed in figure 22-24 (appendix I). For the sake of simplicity we do not name the pieces rather we identify them by their type number which range from 1-6.

We also create a randomly initialized game that has not been passed through the evolutionary process for optimization against entertainment. This game is used to analyze the learnability of the game experiment covered in next section and also in the user survey, to compare with the evolved games. Rules of the random game are shown in figure 25 (appendix I).

8.1.3 Learnability of Evolved Games

The entertainment value of the evolved games needs to be verified against some criteria other than the proposed entertainment metrics. For this purpose we use the Schmidhuber's theory of artificial curiosity [2]. We need to see how quickly a player learns an evolved game. Games learned very quickly will be trivial for the player and thus not contributing anything towards entertainment. Those taking large amount of time to learn will be too difficult. Games between these two boundaries will fall in the range of entertaining games. To observe the learnability of the evolved games there are two options first to ask a human to play a game multiple times and see how fast she/he learns and second is to do the same task using a software based controller.

We have used an ANN based controller. The architecture of the controller is the same architecture used by Chellapilla [21] for evolving an expert checkers player. There are total 5 layers in the ANN, input with 64 neurons, first hidden layer with

91 neurons; second hidden layer with 40 neurons third with 10 neurons and the output layer with 1 neuron. A hyperbolic tangent function is used in each neuron. The connection weights range from [-2, 2]. The training of the ANN is done using co-evolution. A set of genetic algorithm (GA) population is initialized that represent the weight of the ANN. Each individual of the population is played against randomly selected 5 others. Mutation is the only genetic operator used; we have kept the ANN and its training as close to Chellapilla [21] work as possible, except for the number of iteration for which the ANN is trained. We train the set of ANN until we get a set of weights that beats all others. Such individual will have its fitness equal to 1. The number of iterations that take to get such individual in the archive is called the learning duration or learnability of the game. Figure 10 shows the learnability of all the 4 games including the random game (referred to as game 4).

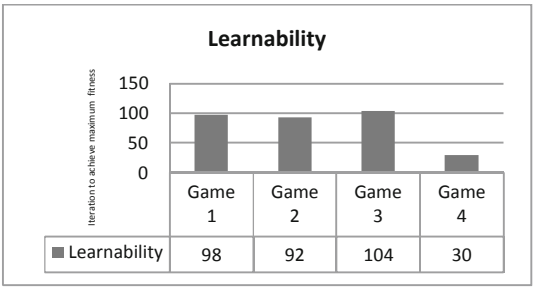

Fig. 10 Learnability of evolved and random game

It takes about 80 to 110 iterations to get a chromosome representing ANN weights that achieve a fitness of 1 during co-evolution, for the evolved games. In case of the randomly initialized game (game 4) it takes only 30 iterations, thus showing game 4 is trivial and uninteresting. Thus we can conclude that game 1, 2 and 3 proof to be entertaining as an ANN based controller neither takes too short a time nor too long to learn these games.

8.1.4 User Survey on Entertainment Value of Evolved Games

To validate the results produced against human entertainment value we have to perform a human user survey. For this purpose we select a set of 10 subjects. Subjects are chosen such that they have at least some level of interest towards computer games. Each individual was given 4 games while s/he was supposed to play each game 3 times, the rules of the game were already explained to the subjects and also displayed on the software they used. This makes a total of 12 games to be played by each subject.

The four games given to the user are marked as Game 1, Game 2, Game 3 and Game 4. Game 4 is the randomly initialized one (but will legal values) whereas remaining three games are evolved for entertainment. Each subject was asked to rank the game they play as 1- liked, 2-disliked and 3-neutral. The results of the human user survey are shown in table 3. For visual purposes we use for √ liked, × for disliked and ~ for neutral.

Table 5 Human user survey results

Subject	Game 1			Game 2			Game 3			Game 4		
	Run 1	Run 2	Run 3	Run 1	Run 2	Run 3	Run 1	Run 2	Run 3	Run 1	Run 2	Run 3
1	~	√	√	~	√	√	~	√	√	~	√	×
2	~	~	√	~	√	√	~	√	√	~	×	×
3	×	√	√	~	×	×	√	×	×	×	×	×
4	~	√	√	√	√	√	√	√	√	√	√	√
5	~	√	√	√	√	√	~	√	√	~	√	×
6	~	√	√	~	√	√	√	√	√	~	√	×
7	~	~	√	~	√	√	√	√	√	~	×	×
8	×	√	√	√	√	√	~	√	√	×	×	×
9	√	√	√	×	√	√	~	√	√	×	×	×
10	√	√	√	×	√	√	~	√	√	~	~	~

For the purpose of collecting statistics from the human user survey we mark a game liked by the subject if during any run he has liked the game and for rest he has either liked or been neutral. If in any run he has disliked the game we mark the game to be disliked. Figure 11 shows the statistics based upon this scheme. About 70- 90 percent subjects have liked the evolved games and found these entertaining, whereas only 10 percent say that the random game (game 4) was entertaining.

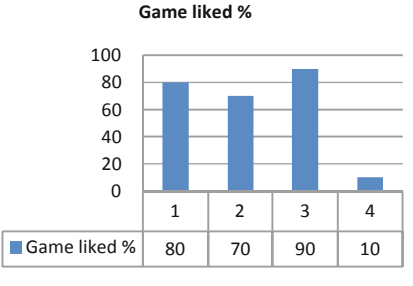

Fig. 11 Statistics of the human user survey

8.2 Predator Prey Games

A population of 10 chromosomes is randomly initialized by the GA. In each generation one offspring is created for each chromosome by duplicating it and then mutating any one of its gene, where all genes have equal probability to be selected. The mutation is done by replacing the existing value with some other permissible value, where each permissible value has equal probability to be selected. This result in a parent and offspring pool of total 20 chromosomes from which 10 best, based on their fitness rank, are selected for the next generation. This evolutionary process is continued for 100 generations.

The fitness of a chromosome, in our case, is based on data obtained by playing the game according to the rules encoded in the chromosome. As there are a number of probabilistic components in the game, hence the data obtained from playing the same game multiple times is never the same. To minimize the effect of this noise in the fitness, we take the average of ten games for every fitness evaluation. To analyze the effect of each of the four proposed factors of entertainment individually we evolve four different populations. Each of these populations is guided by one of the proposed fitness function. We also evolve a population using the combined fitness function.

8.2.1 Duration of the Game

Considering the duration of the game play as determined by the average life span of the agent we rapidly evolve a population of chromosome in which there are very less possibilities for the agent to die in the allotted 100 steps. Figure 12 shows one such evolved chromosome. It is worth noticing in figure 12 (a) that none of the collision logic column 19, 20 and 21 contain 1 which represents death of the agent. Rules of the games in figure 12 (b) shows death of the agent only in case it collides with red type predator but as shown in column 1 there are 0 instances of red predator in the game. Thus these games will be played for a longer duration of time.

	Number of Predators			Movement Logic			Collision Logic														Score Logic								
	R	G	B	R	G	B	R-R	R-G	R-B	R-A	G-R	G-G	G-B	G-A	B-R	B-G	B-B	B-A	A-R	A-G	A-B	R-R	G-G	B-B	A-R	A-G	A-B	G-R	B-R	B-G
(a)	3	17	13	3	0	3	2	1	1	0	0	1	0	2	1	2	2	2	0	0	2	-1	-1	0	-1	0	0	-1	-1	0
(b)	0	18	10	0	1	3	2	2	1	1	2	0	0	1	1	1	1	1	1	0	2	0	1	0	1	-1	-1	1	-1	-1

Fig. 12 The best chromosome evolved by optimizing duration of game. a) Evolved using rule based controller b) Evolved using ANN based controller

8.2.2 Appropriate Level of Challenge

Considering this metric alone, we were able to evolve chromosomes which encouraged a score of 10 to 20. An example is shown in Figure 13. The game rules represented by figure 13 (a) shows that agent looses point only when it collides with blue predators, although agents score also decreases when green/red predator

collides or if there is collision between blue type predators. The score remains neutral when collision occurs between blue/green and agent/red. In all other cases score is increased. Agent does not die out in any of the collision case. The rules represented by figure 13 (b) shows a decrease of agents score in case of red/red collision only. Here agent does not die out in any of the collision case.

	Number of Predators			Movement Logic			Collision Logic														Score Logic								
	R	G	B	R	G	B	R-R	R-G	R-B	R-A	G-R	G-G	G-B	G-A	B-R	B-G	B-B	B-A	A-R	A-G	A-B	R-R	G-G	B-B	A-R	A-G	A-B	G-R	B-R	B-G
(a)	10	0	11	0	0	3	1	2	1	1	0	0	2	0	2	2	2	2	2	0	2	1	1	-1	0	1	-1	-1	1	0
(b)	7	7	8	2	4	3	2	0	2	2	0	0	2	1	2	0	0	0	2	0	2	-1	0	0	0	1	1	1	1	0

Fig. 13 The best chromosome evolved by optimizing appropriate level of challenge. a) Evolved using rule based controller b) Evolved using ANN based controller

8.2.3 Diversity

When the diversity is considered in isolation, the solutions evolved tend to have somewhat higher number of predators of each type. Another tendency is to have one of the dynamic movement logics. Yet another observation was that the death of agent was avoided in most of the better chromosomes so that the game may continue for maximum number of steps. A sample chromosome is shown in Figure 14.

	Number of Predators			Movement Logic			Collision Logic														Score Logic								
	R	G	B	R	G	B	R-R	R-G	R-B	R-A	G-R	G-G	G-B	G-A	B-R	B-G	B-B	B-A	A-R	A-G	A-B	R-R	G-G	B-B	A-R	A-G	A-B	G-R	B-R	B-G
(a)	0	10	0	2	4	2	0	1	2	1	2	0	2	0	1	1	0	2	1	0	2	-1	0	-1	1	0	-1	0	-1	-1
(b)	0	17	0	4	4	4	1	2	0	0	0	1	2	2	1	1	1	2	1	0	2	-1	-1	0	0	-1	-1	-1	1	0

Fig. 14 The best chromosome evolved by optimizing diversity. a) Evolved using rule based controller b) Evolved using ANN based controller

8.2.4 Usability

The chromosomes evolved by using usability of the play area metric seem similar to that for diversity metric. One such chromosome is shown in figure 15. The game rules shown in figure 15 show that usability as an entertainment metrics alone encourages rules with maximum number of predators of each type none having movement logic 0 which means predator does not move.

	Number of Predators			Movement Logic			Collision Logic														Score Logic								
	R	G	B	R	G	B	R-R	R-G	R-B	R-A	G-R	G-G	G-B	G-A	B-R	B-G	B-B	B-A	A-R	A-G	A-B	R-R	G-G	B-B	A-R	A-G	A-B	G-R	B-R	B-G
(a)	20	18	19	1	2	2	2	0	0	2	2	1	2	0	2	2	2	0	2	0	2	1	-1	0	1	0	0	-1	0	0
(b)	19	20	20	4	4	2	2	0	2	2	2	2	0	2	2	2	0	2	2	0	2	-1	-1	1	-1	1	1	0	-1	-1

Fig. 15 The best chromosome evolved by optimizing usability. a) Evolved using rule based controller b) Evolved using ANN based controller

8.2.5 The Combined Fitness Function

The above mentioned combined fitness function was used to evolve a population of chromosome. Most of the resulting evolved chromosomes are playable and seem interesting. They were much better than random games. Also they seem better than some games evolved using individual components of the fitness function. However, an extensive user survey is needed to verify and quantify all these observations, which is covered in next sub section of this chapter. A chromosome showing the best evolved chromosome is shown in figure 16.

	Number of Predators			Movement Logic			Collision Logic														Score Logic								
	R	G	B	R	G	B	R-R	R-G	R-B	R-A	G-R	G-G	G-B	G-A	B-R	B-G	B-B	B-A	A-R	A-G	A-B	R-R	G-G	B-B	A-R	A-G	A-B	G-R	B-R	B-G
(a)	15	8	3	1	1	0	1	2	0	0	1	0	1	2	0	2	2	0	0	0	0	1	1	-1	-1	-1	1	-1	-1	-1
(b)	11	20	3	2	4	2	2	2	0	2	2	2	0	1	1	1	1	2	0	0	2	0	1	1	-1	1	0	-1	0	1

Fig. 16 The best chromosome evolved by optimizing the combined fitness function. a) Evolved using rule based controller b) Evolved using ANN based controller

8.2.6 Human User Survey

To validate the results produced against human entertainment value we performed a human user survey on 10 subjects, chosen such that they have at least some aptitude towards playing computer games. The experiment was conducted in two different sets on different days using the same subjects, one for the games evolved using rule based controller and second for the games evolved by ANN based controller. Each individual was given 6 games while s/he was supposed to play 2 times, the rules of the game were already explained to the subjects and also displayed on the software they used. This makes a total of 12 games to be played by each subject in each set of experiment. The six games given to the user were marked as A, B, C, D, E and F. Where A was a randomly initialized game, B was the game evolved by using duration of game as the fitness function in isolation, C was the game evolved by using appropriate level of challenge as entertainment metrics, D was evolved against diversity, E for usability and F was the game which was evolved based on all the four entertainment matrices combined. The subjects were unaware of criteria of evolution for each game. Randomly selected game rules are shown in figure 17. Each subject was asked to rank the game they play as 1- liked, 2-disliked and 3-neutral.

The randomly generated game, game code A, rules consists of 0 red predators, 12 green predators collision with them cause 0 effect on agents score but causes it

Number of Predators			Movement Logic			Collision Logic														Score Logic								
R	G	B	R	G	B	R-R	R-G	R-B	R-A	G-R	G-G	G-B	G-A	B-R	B-G	B-B	B-A	A-R	A-G	A-B	R-R	G-G	B-B	A-R	A-G	A-B	G-R	B-R	B-G
0	12	0	3	0	2	2	1	0	0	0	1	1	2	2	1	1	1	1	0	1	0	1	1	1	0	1	0	0	-1

Fig. 17 Randomly select game rules for user survey

to die, 20 blue predators and they decreases the agent's score by 1 and also kills the agent when collision occurs. The game rules has the tendency to kill the agent very early and always with 0 or negative scores. The survey suggests that every subject disliked this game.

A pie chart representation of the survey is given in figure 18 and 19. As all the games are played by each subject twice we have marked a game to be liked by a subject if for once s/he has liked it and second time been neutral. If in any of the play subject disliked the game we mark it as disliked. If a subject has been neutral in both iterations of game play it is not considered. In all other cases the game is marked as liked by the subject.

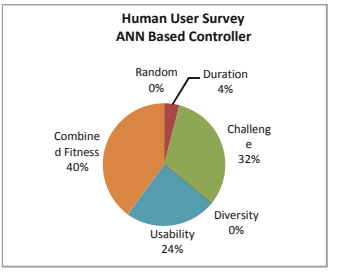

Fig. 18 Pie chart representation of the human user survey using rule based controller.

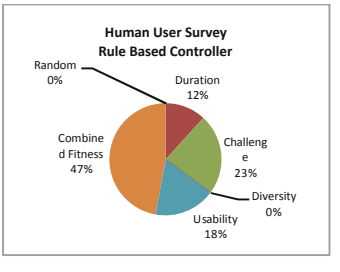

Fig. 19 Pie chart representation of the human user survey using ANN based controller.

The purpose of evolving two sets of population was to see the effect of controller type on the evolved games. Theoretically the rule based controller will evolve the games which are entertaining only while the game is played using the rules encoded in the rule based controller. Thus the idea of using a relatively intelligent and generic controller using an ANN was implemented. Although the games evolved using both the controllers seems somewhat same but in comparison the games evolved using the ANN based controller is more liked by the subjects in the user survey. Figure 20 shows a comparison of the two controllers rating, in general except for the duration metrics games evolved by the ANN based controller are rated higher.

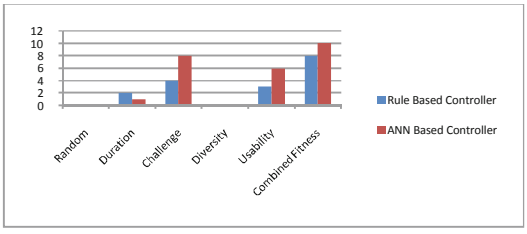

Fig. 20 comparison of the rule based and ANN based controller

8.2.7 Controller Learning Ability

The controller we use is the same ANN based controller we used to evolve the games as shown in figure 8. First of all the weights of the neural network are trained using GA against each of the evolved games and a randomly initialized game, separately.

The learning ability of the controller is calculated as follows:

- The controller plays the game for N times, where N is 10.
- Calculate average score of N games.
- Repeat step 1 and 2 until the standard deviation of the last M runs is minimized. Where M is 3 and the minimized standard deviation is set to 5. We also round the average score to an integer value.

If for 1000 runs condition in step 3 is not satisfied learning ability of the controller is set to 1000. Figure 21 lists the results of the controller learning experiment.

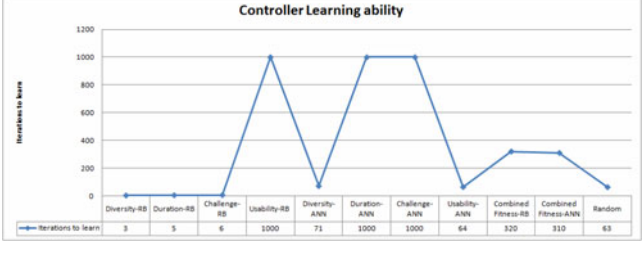

Fig. 21 Learning ability of a controller

The results of the experiments shows that the games evolved using the individual entertainment metrics in isolation were either too easy for the controller to learn, as in case of diversity, duration, challenge evolved using rule based controller and usability and diversity evolved using ANN based controller, or too hard as in case of usability for rule based controller and duration and challenge for ANN based controller. Same goes for the randomly initialized game rules which were learned by the controller in only 63 iterations. In contrast the game rules evolved

using combined fitness function (both based on rule based and ANN based controller) were neither too hard nor too easy and lied between the two extremes.

9 Conclusion

The work presented in this chapter is focused on two primary questions: how we can measure the entertainment value of a computer game and how we can automatically generate entertaining games using computational intelligence techniques. For this purpose we used two different genres of games as our test bed which are, board based games and predator/prey games. Based on different theories of entertainment in computer games we identify different factors that contribute towards entertainment in each genre of game. In the process we have presented some metrics for entertainment which are based on duration of game, level of challenge, diversity, intelligence, and usability. These metrics are combined in a fitness function to guide the search for evolving the rules of the game.

Further work needs to be done to make the proposed entertainment metrics more generic so they can automatically cover more types of computer games. The automatic game creation approach can be applied to other genre of games like platform games, real time strategy games and many others. Some other direction could be to use co-evolution where one population tries to evolve the rules and other tries to evolve the strategy to play on those rules. In the context of entertainment metrics in our case the four fitness criterion are linearly combined, an alternate would be use of multi objective genetic algorithm for this purpose. There may be a study conducted on effect of the type of controller being used for evolution process. It will be interesting to see if different types of controllers producing entertaining yet totally different types of games. The idea of measuring entertainment and automatic generation needs to be extended to other real world applications. One such application could be developing strategies for wars.

References

[1] Olson, C.K., Kutner, L.A., Warner, D.E., Almerigi, J.B., Baer, L., Nicholi, A.M., Beresin, E.V.: Factors correlated with violent video game use by adolescent boys and girls. Journal of Adolescent Health, 77–83 (2007)

[2] Schmidhuber, J.: Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts. Connection Science 18, 173–187 (2006)

[3] Iida, H., Takeshita, N., Yoshimura, J.: A Metric for Entertainment of Board Games: Its Application for Evolution of Chess Variants. In: Nakatsu, R., Hoshino, J. (eds.) Entertainment Computing: Technologies and Applications Proceedings of IWEC 2002 , pp. 65–72. Kluwer Academic Publishers, Boston (2003)

[4] Cincotti, A., Iida, H.: Outcome Uncertainty And Interestedness In Game-Playing: A Case Study Using Synchronized Hex. In: New Mathematics and Natural Computation (NMNC), vol. 02(02), pp. 173–181 (2006)

[5] Retalis, S.: Creating Adaptive e-Learning Board Games for School Settings Using the ELG Environment. Journal of Universal Computer Science, 2897–2908 (2008)

[6] Togelius, J., Nardi, R.D., Lucas, S.M.: Towards automatic personalised content creation for racing games. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games, Piscataway, NJ, April 1-5 (2007)
[7] Yannakakis, G.N., Hallam, J.: Towards Optimizing Entertainment In Computer Games. Applied Artificial Intelligence 21(10), 933–971 (2007)
[8] Gallagher, M., Ryan, A.: Learning to Play Pac-Man: An Evolutionary Rule-based Approach. In: Proceedings of IEEE Congress on Evolutionary Computation, Canberra, Australia, December 8-12 (2003)
[9] Lankveld, G., Spronck, P., Rauterberg, M.: Difficulty Scaling through Incongruity. In: Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference, Stanford, California, October 22-24 (2008)
[10] Togelius, J., Schmidhuber, J.: An Experiment in Automatic Game Design. In: Proceedings of IEEE Computational Intelligence and Games, Perth, Australia, December 15-18 (2008)
[11] Pedersen, C., Togelius, J., Yannakakis, G.N.: Optimization of platform game levels for player experience. In: Proceedings of Artificial Intelligence and Interactive Digital Entertainment, Stanford, California, October 14-16 (2009)
[12] Compton, K., Mateas, M.: Procedural Level Design for Platform Games. In: Proceedings of 2nd Artificial Intelligence and Interactive Digital Entertainment Conference, Stanford, California, June 20-23 (2006)
[13] Yannakakis, G.N., Hallam, J.: Evolving Opponents for Interesting Interactive Computer Games. In: Proceedings of the 8th International Conference on the Simulation of Adaptive Behavior, Los Angeles, USA, July 13-17 (2004)
[14] Csikszentmihalyi, M.: Flow.: The Psychology of Optimal Experience. Harper & Row, New York (1990)
[15] Csikszentmihalyi, M., Csikszentmihalyi, I.: Introduction to Part IV in Optimal Experience: Psychological Studies of Flow in Consciousness. Cambridge University Press, Cambridge (1988)
[16] Malone, T.W.: What makes computer games fun? Byte 6, 258–277 (1981)
[17] Koster, R.: A Theory of Fun for Game Design. Paraglyph Press (2005)
[18] Rauterberg, M.: Amme: An Automatic Mental Model Evaluation to Analyze User Behavior Traced in a Finite, Discrete State Space. In: Proceedings of the Annual Conference of the European Association of Cognitive Ergonomics, EACE 2005, Chania, Crete, September 29-October 1 (2005)
[19] Rauterberg, M.: About a Framework for Information and Information Processing of Learning Systems. In: Falkenberg, E., Hesse, W., Olibve, A. (eds.) Information System Concepts, pp. 54–69. IFIP Chapman & Hall, Boca Raton (1995)
[20] Yannakakis, G.N.: How to Model and Augment Player Satisfaction: A Review. In: Proceedings of the 1st Workshop on Child, Computer and Interaction, ICMI 2008, Chania, Crete (October 2008)
[21] Chellapilla, K., Fogel, D.B.: Evolving an expert checkers playing program without using human expertise. IEEE Trans. Evolutionary Computation 5(4), 422–428 (2001)

Glossary of Terms and Acronyms

ANN: Artificial Neural Network
Chromosomes: One individual of the genetic algorithm population representing one complete game
ES: Evolutionary Strategy
Fitness function: the objective of the evolution process
GA: Genetic Algorithm
Learnability: The duration in number of steps an artificial intelligence based controller takes to learn a game
Piece of honour: A piece in a board based game having the highest priority and whose absence will make relevant player loose the game
Search space: The total scope of the game rules which will be used to evolve new games.
Software agents: the artificial intelligence based controllers to automatically play the game

Appendix I

Piece No	Movement Logic	Step Size	Capturing Logic	Conversion Logic
1	L	Multiple	Step Into	6
2	Diagonal Forward & Backward	Single	Step Over	5
3	All Directions	Multiple	Step Into	Nil
4	Straight Forward	Multiple	Step Into	1
5	Straight Forward	Multiple	Step Over	2
6	All Directions	Multiple	Step Over	3
	Piece of Honour	5		
	Mandatory to Capture	No		

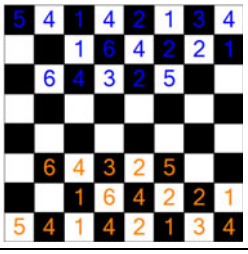

Fig. 22 Rules of game 1 and pieces board positions

Piece No	Movement Logic	Step Size	Capturing Logic	Conversion Logic
1	L	Single	Step Over	1
2	Diagonal Forward	Single	Step Into	3
3	Diagonal Forward	Single	Step Over	Nil
4	All Directions	Multiple	Step Over	Nil
5	Straight Forward	Multiple	Step Into	2
6	All Directions	Single	Step Into	1
	Piece of Honour	5		
	Mandatory to Capture	Yes		

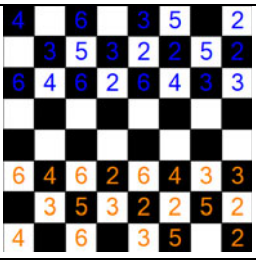

Fig. 23 Rules of game 2 and pieces board positions

Piece No	Movement Logic	Step Size	Capturing Logic	Conversion Logic
1	L	Multiple	Step Into	5
2	L	Single	Step Into	2
3	Straight Forward	Multiple	Step Over	3
4	Straight Forward	Single	Step Over	5
5	Diagonal Forward & Backward	Single	Step Over	3
6	Diagonal Forward	Multiple	Step Over	3
	Piece of Honour	Nil		
	Mandatory to Capture	Yes		

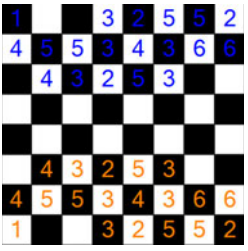

Fig. 24 Rules of game 3 and pieces board positions

Piece No	Movement Logic	Step Size	Capturing Logic	Conversion Logic
1	All Directions	Multiple	Step Into	Nil
2	All Directions	Single	Step Over	1
3	Straight Forward & Backward	Multiple	Step Over	5
4	Diagonal Forward & Backward	Multiple	Step Over	2
5	Straight Forward	Multiple	Step Into	6
6	Straight Forward & Backward	Single	Step Into	6
	Piece of Honour	5		
	Mandatory to Capture	No		

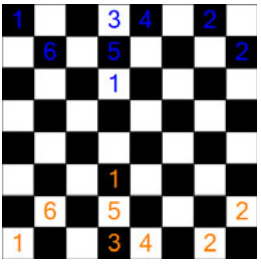

Fig. 25 Rules of random game and pieces board positions