International Journal of Computational Intelligence Systems, Vol. 8, No. 6 (2015) 1128-1143

Computational Intelligence-based Entertaining Level Generation for Platform Games

Zahid Halim
Faculty of Computer Science and Engineering, Ghulam Ishaq Khan Institute of Engineering Sciences and Technology Topi, Pakistan.

Abdul Rauf Baig
Department of Information Systems,College of Computer and Information Sciences, Al Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, Saudi Arabia

Ghulam Abbas
Faculty of Computer Science and Engineering, Ghulam Ishaq Khan Institute of Engineering Sciences and Technology Topi, Pakistan.

E-mail: zahid.halim@giki.edu.pk, raufbaig@ccis.imamu.edu.sa, abbasg@giki.edu.pk

Received 21 November 2014

Accepted 5 October 2015

Abstract

With computers becoming ubiquitous and high resolution graphics reaching the next level, computer games have become a major source of entertainment. It has been a tedious task for game developers to measure the entertainment value of the computer games. The entertainment value of a game does depend upon the genre of the game in addition to the game contents. In this paper, we propose a set of entertainment metrics for the platform genre of games. The set of entertainment metrics is proposed based upon certain theories on entertainment in computer games. To test the metrics, we use an evolutionary algorithm for automated generation of game rules which are entertaining. The proposed approach starts with an initial set of randomly generated games and, based upon the proposed metrics as an objective function, guides the evolutionary process. The results produced are counterchecked against the entertainment criteria of humans by conducting a human user survey and a controller learning ability experiment. The proposed metrics and the evolutionary process of generating games can be employed by any platform game for the purpose of automatic generation of interesting games provided an initial search space is given.

Keywords: Computer Games, Entertainment Computing, Automatic Game Creation, Genetic Algorithms, Interestingness Metrics for Computer Games.

1. Introduction

Computer games have become a major source of entertainment for all age groups, especially children. The gaming industry is expanding at an exponential rate and the reason for computer games becoming the primary source of entertainment range from high resolution graphics to a diverse level of choice and challenge. According to a survey conducted in [1] on 1254 subjects, 93% reported playing electronic games. The results in [1] show the popularity of computer and video games in young generation. However, reports [2] show that the games are equally popular in elderly as they are in children. The average age of a game player in certain geographical areas is 30 years [2].

Game developers face the challenge of measuring entertainment for human users. The complexity lies in the fact that entertainment is a subjective matter. 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, distance, and many such matters are measured. This would give a quantitative representation of the entertainment a game provides. However, based upon the measurable entertainment, the responsibility of producing a game, like today, will still be on the shoulders of game developers. Game developers will have to define the complete game from start till the end along with each stage, and its components and complexities. It would be very convenient if we could also produce the game automatically based upon the measurable entertainment. This would lead to interesting applications in the area of computer game development.

In this paper, we address the aforementioned two issues in game development: measuring entertainment value of a game and automatic generation of entertaining game rules. Our focus will be the platform genre of games. We propose an entertainment metrics to quantitatively measure the entertainment value of the game. The proposed metrics consider four criterion of entertainment including: (i) duration, (ii) challenge, (iii) diversity, and (iv) usability. These four criterions measure the theoretical concept of challenge and curiosity in the game, which play an important role in making a game more entertaining. There might be other sources of entertainment, like graphics and sound effects, but these factors are not in the scope of the basic ingredients of a game. That is the reason why these factors have not been considered while devising the entertainment metrics. We show the utility of the proposed entertainment metrics by generating new and entertaining game rules through an evolutionary process, which devises our entertainment metrics as a fitness function by guiding the evolution towards more entertaining game rules. In order to counter check the entertainment value of the evolved game rules, we have conducted a human user survey and a controller learning experiment, where the results correlate with those produced by the system.

The rest of the paper is organized as follows. Section 2 covers the previous work done in the context of measuring entertainment and automatic game generation. Section 3 lists the entertainment theory based upon which we have proposed our entertainment metrics. Section 4 introduces the platform genre of games for which we have proposed the entertainment metrics. Section 5 lists the search space we have used to evolve the game rules. Section 6 explains the chromosome encoding of the rules. Section 7 explains the entertainment metrics and the fitness function. Section 8 is about the controller, which is a rule based one, to play the games. Section 9 contains the experiments and their results. Section 10 lists the methods to verify the results produced by conducting a human user survey and a controller learning ability experiment. Finally, Section 11 concludes the paper along with presenting the future directions.

2. Literature Survey

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. Iida et al. [3] propose a measure of entertainment for games and use it to analyze the evolution of chess over the centuries. This measure is considered to be the pioneer in quantification of entertainment in games. Even though Iida's work is limited to chess and its variants, the measure of entertainment can also be 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/her turn. The game is considered more entertaining if the value of Iida's measure is low. In [4], the authors introduce the uncertainty of the game outcome as a metric of entertainment. If the outcome is known at an early stage, then there is not much interest in playing the game. Similarly, if it is found at the last move, then it is considered probabilistic. The outcome should be unknown for a large duration of the game and it should become known in the last few moves of the game. The authors state that it is easy to create new board games and variants of classical games, but making these attractive to the human users is challenging. In [4], a technique based on synchronism and stochastic elements is used to refine the game of Hex. Symeon et al. [5] use board games for e-learning. The work in [5] proposes an e-learning board game that adopts the basic elements of a racing board game and cultivates skills like creativity, problem-solving, and imagination in students. The issues of measuring the entertainment value of the games and automatic generation of game contents are not addressed in [5]. Togelius et al. [6] have presented an approach to evolve entertaining car racing tracks for video games. Tracks were represented as b-splines and the fitness of a track depended on how an evolved artificial neural network (ANN) based controller performed on the track. The game model used for experimentations in [6] is 2D. The work in [7] proposes three metrics (which are combined

into one) 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 averaged over multiple runs. The metric *T* 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 (*S*). It is the standard deviation of a player's lifetime over multiple runs having a high value if there is diversity in opponent's behaviour. The third metric is spatial diversity metric $E\{H_n\}$, which is the average entropy of grid-cell visits by the opponents over multiple runs. Its value is high if the opponents move all the time and cover the cells uniformly. The three metrics are combined into one metric $I = [\gamma T + \delta S + \varepsilon E\{Hn\}]/[\gamma + \delta + \varepsilon]$, where *I* is the interest value of the predator/prey game, and γ, δ, and ε are weight parameters. The works in [8,9,10] are somewhat extensions of [7]. In [11], the authors have developed a computer game called *"Glove"* with three levels of incongruity: hard, easy, and balanced. The assumption is that the player would get frustrated or bored, with the first two settings and would enjoy with the third one. However, the verification of this assumption needs to be done. Authors in [11] 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 [12], an effort has been made to evolve rules of the game. The evolution of games in [12] is guided by a fitness function based on the "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. They employ the theory of artificial curiosity-based fitness function introduced in [13], which focuses on the predictability of the game environment. In [14], the authors identify how the *Super Mario* game works. It defines the rules and highlights the goals of the game that the user must reach. More importantly, it shows how the content is generated in the game. According to Nicole et al. [15], people play games to change or structure their internal experiences. Adults enjoy filling their heads with thoughts and emotions unrelated to work or school, others enjoy the challenge to test their abilities [15]. Games offer an efficiency and order in playing that they want in life.

Kate et al. [16] believe that the level design in platform games relies on rhythm. When a player is in rhythm of the game, making jumps requires not only correct distance calculation but also timing. When obstacles are placed in rhythm, they make the movement of players rhythmic, hence, making each jump easier. In [17], a methodology for optimizing player satisfaction in games on a physical interactive platform, known as *Playware,* is demonstrated. An ANN is used to map individual playing characteristics to suggest entertainment preferences for the game players. Based on the preferences, controllable game parameters are adjusted in real-time in order to improve the entertainment value of the game for the player. Performance of the mechanism is evaluated using a survey. In [18], the authors introduce a co-evolutionary system based on a classifier for strategy developments in match-up games. The work in [19] constructs a co-evolutionary system that develops players' strategies. Baghdadi et al. [20] present a procedural content generation approach for Spelunky. Shaker et al. [21] perform a playability check for physics based games. Cardamone et al. [22] present an online tool to generation tracks for two open-source 3D car racing games. Halim et al. [23] present an automated approach for board based games creation using an evolutionary algorithm. Other somewhat related studies using same technique are covered in [9,10,24,25,26].

3. Entertainment Theories and Factors

There are many theories on entertainment in computer games. According to Csikszentmihalyi's theory of flow [27,28], the optimal experience for a person is when s/he is in a state of flow. In this state, the person is fully concentrated on the task that s/he is performing and has a sense of full control [29,30]. 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 [28] proposes two more factors that make games engaging, namely, fantasy and curiosity. If a game has the capability of evoking the player's fantasy and makes the player feel that s/he is somewhere else or doing something exotic, then that game is more enjoyable. Curiosity refers to the game environment. The game environment should have the right amount of informational complexity: novel and comprehensible [31]. Koster's theory of fun [32] 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 [33,34] introduce 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 that a person has of the task. We have a positive incongruity if this difference is positive, and a negative incongruity otherwise. In case of a 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 too negative. In case of a large positive incongruity, the humans have a tendency to avoid the task and, in situations of a large negative incongruity, they get bored. This requirement of the right amount of incongruity is similar to the right amount of challenge in the concept of flow mentioned earlier. Malone [35] describes in his work the factors that make a game more fun to play. Challenge, curiosity, and morals are the three key factors that influence game entertainment. The work in [35] also provides various design guidelines and heuristics for the game developers to produce entertaining games.

Koster theory [32] states that when players start playing a game, at the very beginning it might seem to be fine but after sometime or may be after a long interval of playing, one becomes bored and might shift to another game. After a while, the same happens with the second game as well. In order for a game to be interesting, it must be diverse and the patterns in the games must not reoccur; otherwise the game will be boring for the human players. Today, time stands as an important factor. People need to optimize their utilization of time in different tasks. If the duration of a particular task exceeds a specific threshold, it becomes either boring or a source of mental stress for the user, depending upon the nature of the task. People tend to like the tasks with an appropriate level of challenge. If a task is challenging enough that it cannot be achieved, this results in de-motivation and the person assigned that task pays very little or no effort to achieve the task. Usability is a factor that plays an important role in the interestingness of a game. Consider a game where, during entire duration of the game, the player and the opponents remain in a limited area of the game. Certainly such a scenario is boring. This is one of the reasons, other than many, that levels are introduced in many games to explore different areas of the game. Games with higher level of usability of the play area will certainly be more interesting than those with lower level of usability.

Based upon the above discussion, we have identified four factors that influence the interestingness of a game. These factors include: i) duration, ii) challenge iii) diversity, and iv) usability. Table 1 lists the previous work and the theories based on which we have identified these four factors that influence entertainment value of a computer game. There are many other factors that may contribute towards the interestingness of a game including the light effect, sounds, and graphics. But these factors are not the basic ingredient of a computer game and that is the reason we do not consider them in our metrics of game entertainment. After listing the entertainment factors, the next task is how to measure these four factors for a given game. We formulate equations for this purpose in Section 7 of the paper.

Table 1

Aspects mentioned in literature and theories for entertaining games

Aspect	Previous works	Theories
Game duration	8, 9, 10, 17	15, 32, 35
Challenge	8, 9, 12, 17	29, 33, 34, 35
Diversity	8, 9, 10, 12, 30	28, 32, 35
Usability	8, 9, 16	27, 28

4. Platform games

Platform games stand as a complicated genre of games. A true and universally known representation of platform games is the *Super Mario Bros*. The public domain version of this game, by Markus Perrson called *Infinite Mario Bros*[‡], acts as an ideal test-bed for our research, given the wide variety of objects in the game and the mass appeal of the game. The first ensures a rich environment that enables experimentation on different levels while the latter assists with the measuring of entertainment part. The environment consists of both static and dynamic objects spread across the level and *Mario* travels from left to right on a 2-dimensional screen. Different obstacles come in the way and the goal is to reach the end of the level. The *Infinite Mario Bros* already hosts a mechanism to generate infinitely many random levels for a player. The typical environment of the *Infinite Mario Bros* is shown in Fig. 1. A better approach would be to effectively calculate the entertainment value for one level, and generate the next level based on that. In this work, we propose a solution to the dynamic content generation for the platform games. For this purpose, we introduce a set of metrics to calculate the entertainment value of the games and utilize this metric as fitness function to optimize a set of genetic algorithm (GA) population for entertainment. The choice of GA is made for game content generation due to multiple relevant reasons. GA can represent the contents of the game as a one dimensional chromosome, which makes it convenient to process due to its time and space complexity. A recent survey on search based procedural content generation for computer games [36] also shows the popularity of GAs in game content generation. There exist other optimization approaches like combinatorial optimization, dynamic programming, gradient methods, and stochastic optimization, but these are mostly for function optimization and cannot be used for game content generation. However, genetic programming (GP) can be used as an alternate to GAs, but the GP model represents an individual solution using a tree like structure which will cause the evolutionary process to further slowdown.

Fig. 1. A typical environment of the game *Infinite Mario Bros*

5. Search space

A game is a set of certain entities that are governed by game rules, where people or teams compete against each other. New games cannot be generated and, thus, not evolved out of nowhere unless a basic set of search space is provided based upon which the new games can be created. The search space will provide all the possible type of entities, their number, and the rules they will be governed by. To implement an evolutionary algorithm for game content generation, we need to understand the basic search space of our model game - *Infinite Mario Bros*. The environment consists of a standard screen covering *15 x 15* cells. The 2-dimentional controller character *Mario* can walk/run in forward and backward directions. It can also jump and shoot fireballs. Holes and moving enemies hinder *Mario's* goal of finishing the level. A level ends if *Mario* reaches the goal or dies during the process. Certain items are scattered around the stage, either visible or hidden in bricks, and need to be collected by *Mario* to gain extra score.

6. Chromosome Encoding

As mentioned earlier, for the purpose of evolving the game rules, we have used GA. Each chromosome of the GA population represents one complete set of rules for the game; whereas each gene of the chromosome represents one rule of the game. Based

[‡] https://github.com/cflewis/Infinite-Mario-Bros
https://infinitemariobros.codeplex.com/
https://code.google.com/p/infinite-mario-brs/

upon the aforementioned search space, the structure of the chromosome is listed in Fig. 2. In our version of *Infinite Mario Bros* represented as a chromosome, we have 28 genes. These genes represent *Mario's* size, game difficulty level, time, map type, obstacles, and various enemy types in each chromosome. Gene 1 represents the difficulty level of the game and it may have values form 0 to 10, where 10 is for the maximum difficulty level. The second gene represents the time and may have values between 0-100. Map type is represented by the 3rd gene having values between 0-2, where 0 is for underground, 1 for over ground, and 2 for castle. Genes 4-8 represent cannons, jumps, tubes, hill straight, and straight, respectively. Genes 4-8 can have values between 0-40 representing the occurrence probability of each of these items in the game map. Gene 9-28 represents the 20 enemies in the game and can have value from 0-20 one for each enemy, where 0 stands for no enemy of a particular type.

7. Fitness Function

Entertainment is the key to success while making any game. Each level of the game must be engaging for the user and shall grab the user's full attention. Having said this, a regular platform game might get monotonous by the random or specific rule-based content generation. It will only maintain its attraction until the user finds something new in the various game levels. If the game is not intelligently evolved, it is likely to have a shorter life span. As already mentioned, GAs evolve through their population of chromosomes. The fitness function needs to ensure that the correct factors are considered when ranking the population. In our case, the individual factors of the fitness function include the duration of the game, the level of challenge, the diversity, and the usability of the static and dynamic components.

7.1. Duration of the Game

Duration is an extremely important factor. A game must not end when the user's entertainment graph is rising because the continuous increase highlights more potential entertainment. At the same time, a game must end before the entertainment graph begins to fall so as to maintain the peak entertainment level. In our model game, a regular level ranges from 150 to 250 seconds, if completed. In case *Mario* dies somewhere in the middle of the level, the game ends. The fitness function must ensure that the level of difficulty is not such that a user is unable to reach the end for several turns. If a level ends prematurely too many times, the user is likely to find the game annoying. We calculate the duration D, of the game using equation (1).

$$D = \frac{\sum_{k=0}^{n} L_K}{n}, \quad (1)$$

where L_K is the life of the game playing agent in game number K, and n is the total number of games played. In our experiments, we have used n=10. Given the many probabilistic factors in the game, the duration is calculated by taking an average of n games played. Based on the raw value of D, we scale it in a specific range to reward games with intermediate duration using the ranges given in Fig. 3.

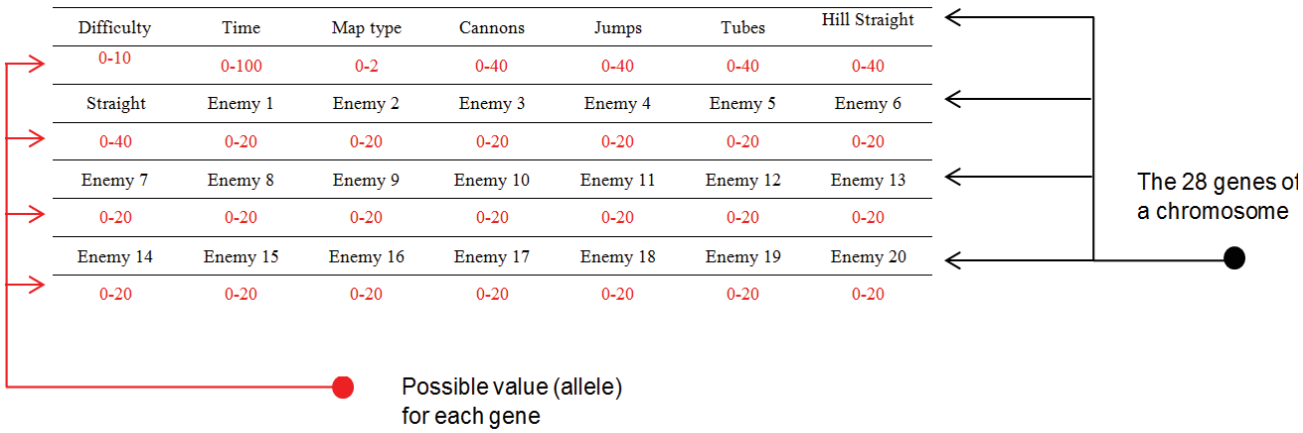

Fig. 2: chromosome encoding for platform games

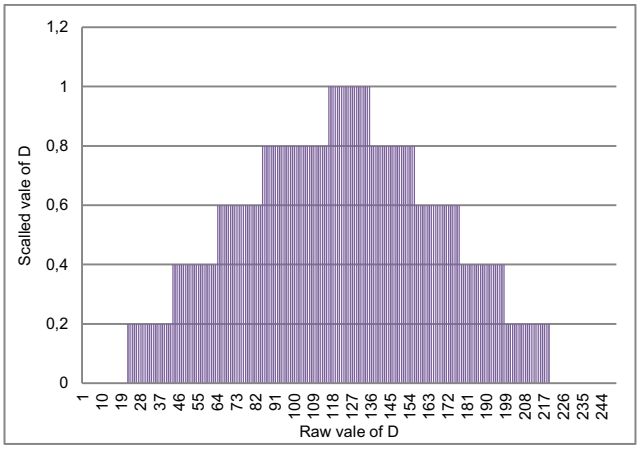

Fig. 3: Scaling ranges for normalized duration

7.2. Appropriate Level of Challenge

If a game is too difficult to play, a player is likely to quit playing. On the other hand, users do not enjoy games that lack any element of challenge. The only option is to provide the correct balance that would keep the user involved. A smooth and long learning curve seems to be an ideal fit for our fitness function [8]. Koster's model rightly identifies the need for challenge in the form of learning, but that too is limited to the user's ability to learn within the given time frame. No learning would represent no challenge, hence, killing the entertainment or any sort of engagement. In our model game, we calculate the score based on three parameters, namely, total distance travelled, total coins collected, and total enemies killed. The three values are added to reach the total score of a player. This leads to a formula to measure the appropriate level of challenge as given in equation (2).

$$c = e^{\left(\frac{-|S_m - S_a|}{S_m}\right)}, \tag{2}$$

where,

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

S_K is score of the agent in game number k,
S_m is the maximum possible score, and
n is the total number of games played which is set to 10, as explained previously.

In this case, the fitness function returns a higher value in the vicinity of the maximum score and gradually decreases for higher or lower scores.

7.3. Diversity

Diversity further spices up the entertainment of a game. Predictable behavior leads to a monotonous game play, which is never appreciated. The fitness function must ensure that the movement of the different objects is sufficiently diverse and holds an element of surprise for the player. To do this, the number of cell changes for all the moving objects can be stored in a data structure. We can average the value achieved by playing a game multiple times. The repetitive behavior of any moving object can be noted and fixed. Equation (3) lists how we measure the diversity of a game.

$$Div = \frac{\sum_{i=1}^{n}\left(\sum_{k=0}^{m}(\partial_k)\right)}{n}, \tag{3}$$

where,

m is the total number of artifacts specified in a chromosome,
∂_k is the number of cell changes made by an artifacts k during a game, and
n is the total number of games played which is set to 10, as explained previously.

7.4. Usability

Usability ensures that the objects are spread out across the complete paly area. If parts of the game level and objects go untouched, the game can be classified as poorly evolved. It shows the wastage of available game resources. It would directly hurt the entertainment factor of the game. A counter can be placed for every cell in the game. It can be incremented when any object touches that cell. Thus, a cell that is never visited during a game will end up with a value of zero. Usability of a game is calculated 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 the usability counter value for a cell k,
$|C_u|$ is the total number of usable cells, and
n is the total number of games played which is set to 10, as explained previously.

7.5. Combined Fitness

The overall fitness function can, therefore, be represented as a sum of the four values mentioned above. To influence the weight of each factor, we multiply each of these factors with a constant value. The evolved formula would then be:

$$FF = aD + bC + cDiv + dU. \quad (5)$$

The value for *a*, *b*, *c*, and *d*, are set to 1 in our experiments. However, we have experimented with various values of these weights in section 10.

8. The Controller

GA uses a population of candidate solution and all these solutions are tested for fitness as the GA iterates. It is impossible to manually check the fitness function value for all the chromosomes. The reason for this is the time factor and human errors, which will result in delayed and inconsistent results. Hence, an automatic controller is required to play as the agent. Such a controller can be implemented as a rule based controller or an intelligent agent. In our case, we have used a rule based controller. 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 controller is composed of rules formulated to implement the following policy. In our case, the rule-based controller has a set of commands. It analyzes the search space ahead of itself and makes the decision accordingly. The agent jumps if an obstacle is detected. It jumps if an enemy is detected and shoots a bullet at the same time. Any coins in the course of action are collected and the time for the game play is recorded. These values are then used by the fitness function. Fig. 4 presents the proposed algorithm (RbC), which the agent uses to play and evaluate the game represented by a chromosome.

9. Experimentation and Results

For our experiments, we are generating 10 chromosomes using the search space. The fitness function ranks these chromosomes, based on the criteria defined earlier. Crossover and mutation are the genetic operators applied on the population. Mutation is applied with a probability of 10%. The GA is run for 1000 iterations (or until no further improvement in the solution is observed). In each iteration, 10 offsprings are created using two random parents, thus, resulting in a parent child pool of 20 chromosomes. From this pool, 10 best chromosomes are promoted to the next population based on their fitness rank. We keep an archive of 5 slots, which is used to store the best chromosome based upon each of the four individual fitness criterions and the combined fitness function. For our analysis, we compare the games evolved using the individual criteria with the best ranked chromosome (based on each of the 4 metrics and the combined fitness function) of the final population and a randomly initialized chromosome. The convergence graph of the GA is shown in Fig. 5, where *C1-C10* represents the 10 candidate solutions. The rules of the game using duration, challenge, dynamics and usability are listed in Fig. 6. Fig. 6 also displays the rules of best game evolved using the combined fitness function and the randomly initialized chromosome.

Algorithm. Rule based controller for the platform game (RbC)

Input: Environment variable form the simulator
Output: Next direction and/or action

```
1:  Pos. x,y = Store current location of the Mario
2:  Distance-Array = Calculate and store distance of Mario from all objects
3:  While ( ! killed ) do
4:      Forward key always true
5:          Analyze search space ahead
6:          If enemy detected, then get the distance and fire and jump
7:          If obstacle detected then analyze the jump needed and jump
8:          If bricks are detected than jump and break bricks
9:  End While
10:  Pos. x,y = 0
11:  Distance-Array = 0
12: return score
```

Fig. 4: Algorithm to play and evaluate a game by a software agent

10. Verification of the results

The games evolved based upon the proposed metrics of entertainment needs to be verified against the entertainment value of the human user. For this purpose, we adopt a twofold strategy. First, we conduct a human user survey of the games evolved by our system. Secondly, we study the learning curve of an ANN based controller on these evolved games and a random games. This gives us a clear picture as to how entertaining the evolved games are. Discussion on both follows.

10.1 Human User Survey

To validate the results produced against the human entertainment value, we performed a human user survey on 20 subjects, which are chosen such that they have at least some aptitude towards playing computer games. The six games presented 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 metrics combined. The subjects were asked to play each game twice before ranking them as 6:very much liked, 4:liked, 2: neutral, and 0:disliked. Table 2 lists the results of the user survey. The results suggest that majority of the subjects, i.e., 86.67% have liked the games evolved using the final fitness rank, whereas only 18.33% liked the randomly initialized game. The hypothesis testing of the human user survey results is listed in section 10.4.

10.2 Controller learning ability

In contrast to the user survey, the entertainment value of the evolved games can be verified using the Schmidhuber's theory of artificial curiosity [13]. 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 entertaining. Those taking longer duration of time to learn will be too complex for the player and will have no or very low entertainment as well. Games between these two extremes will fall in the range of entertaining ones. To see the predictability (which will show how fast a player learns the game, i.e., learning ability) of the evolved games, there are two options: a) ask a human to play a game *N* times and see how fast s/he learns; and b) do the same task using a controller. We have chosen the second option as human verification takes time and also introduces noise in shape of human errors, inaccuracies, and variations. The controller we use is based on an ANN, which is a multi-layer, fully connected feed forward neural network, architecture as shown in Fig. 7. There are a total of 6 neurons in the input layer, 5 neurons in the hidden layer, and 4 output layer neurons. The Sigmoid activation function is used in each neuron. The weights on the edges range between -5 to +5. The input vector to the neural network is (Δx_f, Δy_f, Δx_b, Δy_b, Δx_t, Δy_t) where Δx_f, Δy_f, Δx_b, Δy_b, Δx_t, and Δy_t represent agent's distance in x-coordinate from the nearest object in the forward direction , agent's distance in y-coordinate from nearest object in the front direction, agent's distance in x-coordinate from the nearest object in the backward direction , agent's distance in y-coordinate from nearest object in the backward direction, agent's distance in x-coordinate from the nearest object in the top direction, agent's distance in y-coordinate from nearest object in the top direction, respectively. The ANN outputs a 4 dimensional vector (N_u, N_d, N_l, N_r) where N_u, N_d, N_l, and N_r represent agent's next position for up, down, left, and right, respectively. The agent moves in the direction having the maximum value. For the purpose of training the ANN weights, we have employed GA. Each chromosome of the population represents the set of weights for the entire ANN. A chromosome consists of 97 genes. Mutation is the only genetic operator used. 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 100 iterations and the best chromosome per iteration is saved. As the GA finishes, we select the best chromosome out of 10, based upon the highest average score achieved.

The learning ability of the controller is calculated as follows: (1) the controller plays a game *P* time, where *P* is 10, (2) Calculate average score of *P* games, (3) Repeat step 1 and 2 until the standard deviation of the last *Q* runs is minimized, where *Q* is 3 and the minimized standard deviation is set to 5. We also round-off the average score to an integer value.

If for 1000 runs, the condition in step 3 is not satisfied, the learning ability of the controller is set to 1000, which is considered to be the maximum (indicating a non-entertaining game). The same games

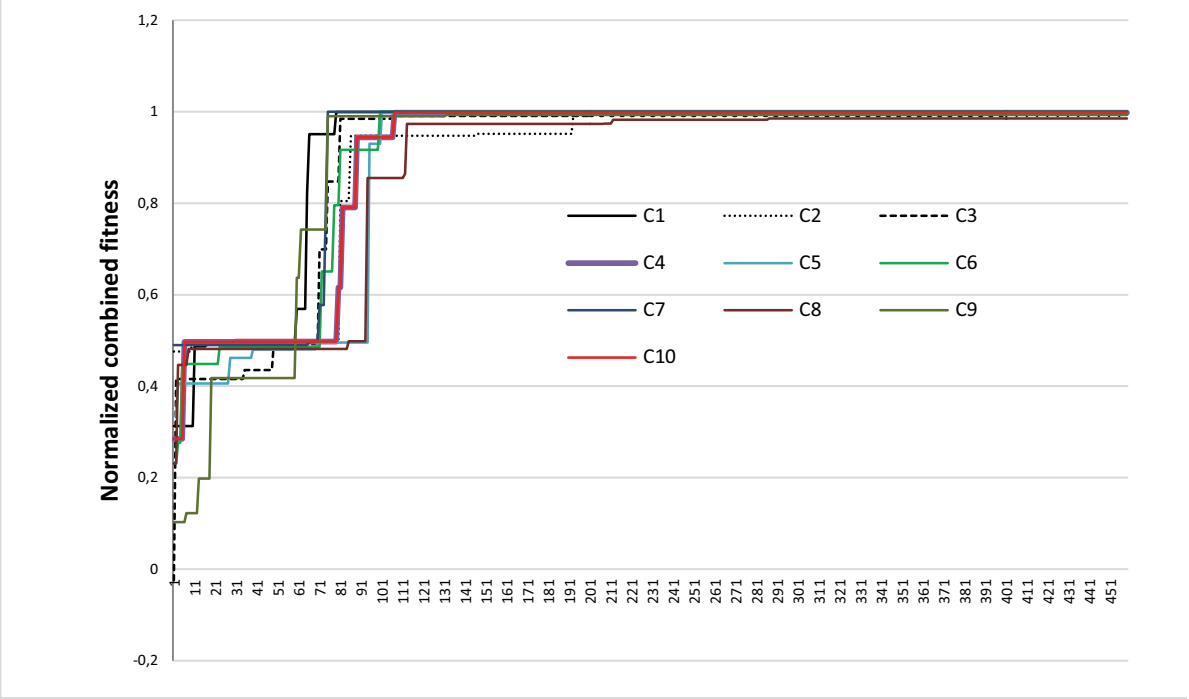

Fig. 5 GA convergence graph for 451 iterations

Game evolved for	Difficulty	Time	Map type	Cannons	Jumps	Tubes	Hill Straight	Straight	Enemy 1	Enemy 2	Enemy 3	Enemy 4	Enemy 5	Enemy 6
Duration	2	98	2	5	3	2	10	20	0	2	0	0	0	2
Challenge	7	84	2	3	4	3	0	20	0	1	3	2	3	1
Diversity	2	80	2	3	4	3	20	0	1	3	2	3	1	1
Usability	4	65	1	5	3	2	10	20	2	2	0	0	0	1
Combined	5	65	2	5	3	2	10	20	2	2	0	0	0	2
Random	1	80	2	1	4	20	12	21	1	1	1	1	2	1

Game evolved for	Enemy 7	Enemy 8	Enemy 9	Enemy 10	Enemy 11	Enemy 12	Enemy 13	Enemy 14	Enemy 15	Enemy 16	Enemy 17	Enemy 18	Enemy 19	Enemy 20
Duration	3	0	2	2	3	2	1	1	3	1	3	2	1	2
Challenge	1	0	3	1	0	1	1	3	3	3	3	2	1	3
Diversity	0	3	1	0	1	1	3	3	3	3	2	1	3	0
Usability	2	3	4	2	2	2	1	2	1	0	3	2	1	1
Combined	3	0	2	3	3	2	1	1	3	1	3	2	1	2
Random	1	1	2	3	0	0	0	1	0	1	0	0	1	0

Fig. 6 Rules of the evolved games using GA and the randomly initialized one

Table 2

Human user survey results

	Game Code					
Subject ID	A	B	C	D	E	F
1	0	4	4	2	4	6
2	0	4	4	4	4	6
3	2	2	4	2	4	6
4	2	4	2	2	2	4
5	0	2	4	2	2	4
6	0	4	2	2	2	4
7	0	2	4	4	4	6
8	0	4	6	4	4	6
9	2	2	2	2	2	4
10	2	2	2	2	2	4
11	0	2	0	2	4	6
12	4	4	4	4	4	6
13	0	2	4	4	4	6
14	0	4	4	2	4	6
15	2	2	6	4	4	6
16	4	0	6	4	4	6
17	0	4	2	2	4	4
18	0	4	2	2	2	4
19	2	2	4	0	4	6
20	2	2	4	0	4	4
%-liked	18.33	46.67	58.33	41.67	56.67	86.67
Average	1.10	2.80	3.50	2.50	3.40	5.20
Median	0.00	2.00	4.00	2.00	4.00	6.00
Std. dev.	1.37	1.20	1.57	1.28	0.94	1.01

A-Random

B-Duration

C-Challenge

D-Diversity

E-Usability

F-Combined

6-Very much liked

4-Liked

2- Neutral

0- Disliked

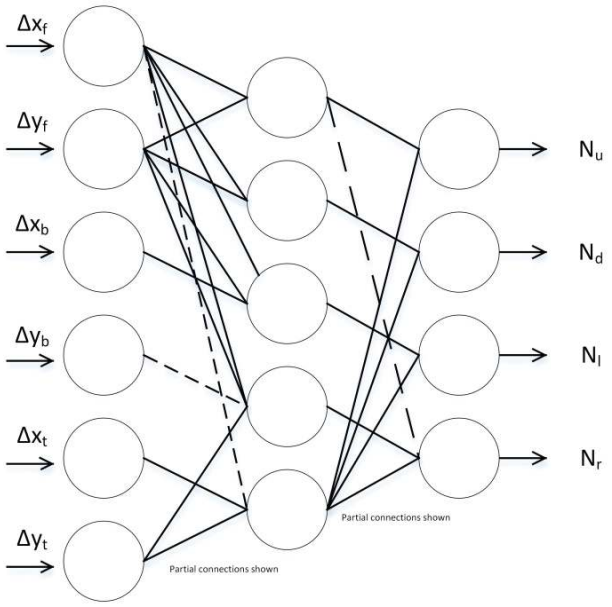

Fig. 7 ANN based controller architecture

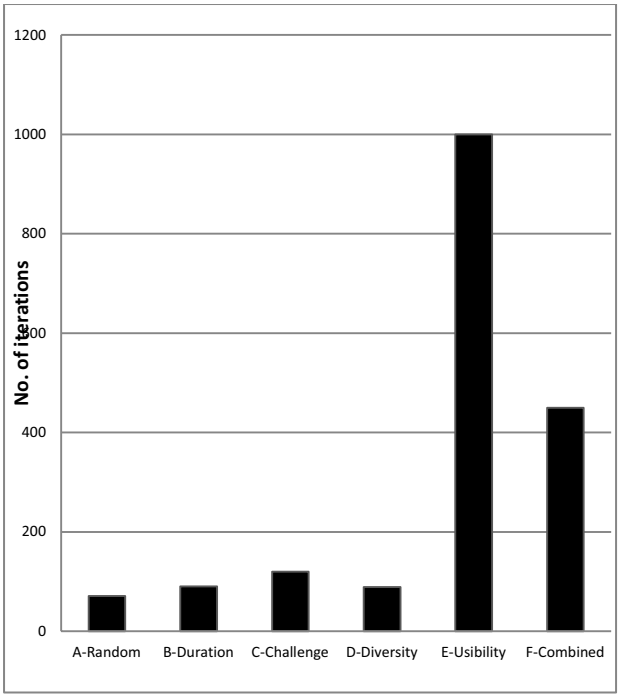

Fig. 8 Controller learning ability for 6 games

Table 3

Analysis of weight in the combined fitness function

	Duration	Challenge	Diversity	Usability	Combined
Duration	87	190	200	1000	
Challenge	190	101	300	201	
Diversity	200	300	97	50	
Usability	1000	201	50	1000	
Combined					450

		Diversity						
		0	0.2	0.5	0.8	1		
Duration	1	1000	1000	1000	1000	**450**	1	Usability
	0.8	90	1000	1000	**455**	**403**	0.8	
	0.5	44	708	**400**	88	**430**	0.5	
	0.2	33	**600**	47	90	88	0.2	
	0	N/A	33	40	30	20	0	
		0	0.2	0.5	0.8	1		
		Challenge						

Fig. 9 Weight analysis for a, b, c, and d

A, *B*, *C*, *D*, *E*, and *F* are used for controller learning ability experiment as were in case of human user survey. The results of the controller learning experiment are shown in Fig. 8. The results of the experiments show that the games evolved using the individual entertainment metrics were either too easy for the controller to learn, as in the case of games evolved against diversity, duration, and challenge, or they were too hard as in the case of usability. Same goes for the randomly initialized game rules, which were learned by the controller in only 71 iterations. In contrast, the game rules evolved using combined fitness function was neither too hard nor too easy and fell between the two extremes.

10.3 Analysis of the weights

To study the effect of weights in the combined fitness function (Eq. (5)), we evolve various games using the GA by setting the value of weighs to 0 and 1. We first set the weight *a* to a maximum (i.e., 1) and set the values of *b*, *c*, and *d* to minimum (i.e., 0). The same procedure is repeated for weight *b*, *c*, and *d* by setting their values to maximum and the rest to minimum. This process gives us 4 games. We also evolve games by setting a combination of two weights (e.g., *ab*, *ac*, *ad*) to maximum and the remaining two to a minimum. This gives us six additional games making a total of 10 games evolved using various combinations of weights in Eq. (5). These 10 games are to be compared against the game evolved using all the four factors having equal (maximum) weight, i.e., game *F* (Fig. 8). Since the manual inspection is time consuming and may also introduce inconsistencies, we use the controller learn-ability approach over these 11 games. Table 3 shows the results of this experiment, where the value in each cell shows the number of iterations the ANN controller takes to learn the game using the corresponding cell's vertical and horizontal headers set to the maximum.

The results in Table 3 show that the games evolved using individual weights set to highest value in isolation are either too trivial or too complex for the ANN controller, thus, they have low entertainment. The same goes for most of the games evolved using the combination of weights with the exception of diversity-challenge and diversity-duration combinations. Overall, the game evolved using all the weights set to an equal value has the most suitable controller learnability value making it the most entertaining game. The weights in Eq. (5) can be useful in creating custom game rules by setting any combination of the weights to higher values as compared to the rest.

The results shown in Table 3 are based on the selection of one or two of the game evaluation criterion out of the available four criterion in Eq. (5). For this analysis, the weights are set to either 0 (minimum value) or to 1 (maximum value). However, Eq. (5) can be used to create many games with weights of *a*, *b*, *c*, and *d* set to any value between 0 and 1. There can be an infinite combination of weight settings for the values in Eq. (5). In order to further study the impact of these weights, we evolve 24 more games. These games are evolved by setting the weights of *a*, *b*, *c*, and *d* to 0, 0.2, 0.5, 0.8, or 1. The aforementioned five weights are representative of the complete spectrum of weights in the range between 0 and 1. As is done for the binary weight analysis in Table 3, the controller learning experiment is repeated for the 24 games evolved with various combinations of weights having values 0, 0.2, 0.5, 0.8, or 1. The results for this experiment are listed in Fig. 9, where the major x-axis lists the values of the weights for *challenge*, minor x-axis lists the values of the weights for *diversity*, the major y-axis lists the values of the weights for *duration*, and minor y-axis lists the values of the weights for *usability*. The value in each cell represents the number of iterations the ANN-based controller requires to learn the game with the weights set to the values represented by the major and minor x-axis, and major and the minor y-axis.

The results in Fig. 9 show that the game evolved with all the weights set to 1 requires 450 iterations by the controller for learning. For cases where the controller requires more than 1000 iterations for learning, we set the value to 1000 indicating an unentertaining game. A value in the range of 400-600 represents the suitable number of iterations required by the controller to learn the game. The games which require the number of iterations in the aforementioned range are classified as entertaining games. Others are classified as unentertaining: either too hard (requires more than 600 iterations) or too trivial (requiring less than 400 iterations). The results in Fig. 9 suggest that if all the weights are set to an equal value, an entertaining game is produced. We also get entertaining games for the weight setting of 0.8, 1, 0.8, and 1, and 0.5, 1, 0.5, and 1 for *the duration*, *challenge*, *usability*, and *diversity*, respectively. Other than evolving entertaining games, these weights can be used to evolve custom flavors of *duration*,

diversity, *usability*, and/or *challenge* in the game rules.

10.4. Hypothesis testing

To statistically prove that the game evolved using the combined fitness function is more entertaining as compared to the other five games, hypothesis test has been conducted using the human user survey data. We devise our null hypothesis (H_0), and the alternate hypothesis (H_1) as follows:

H_0: Game evolved using the combined fitness function is not entertaining as compared to other evolved games.
H_1: Game evolved using the combined fitness

Table 4
Hypothesis test

	One-tailed Student t-test		
Game *F* against	t	p-value	df
Game *A*	10.78	2.05E-13	38
Game *B*	6.97	1.85E-08	38
Game *C*	4.09	0.000114	38
Game *D*	7.43	3.26E-09	38
Game *E*	5.85	4.60E-07	38

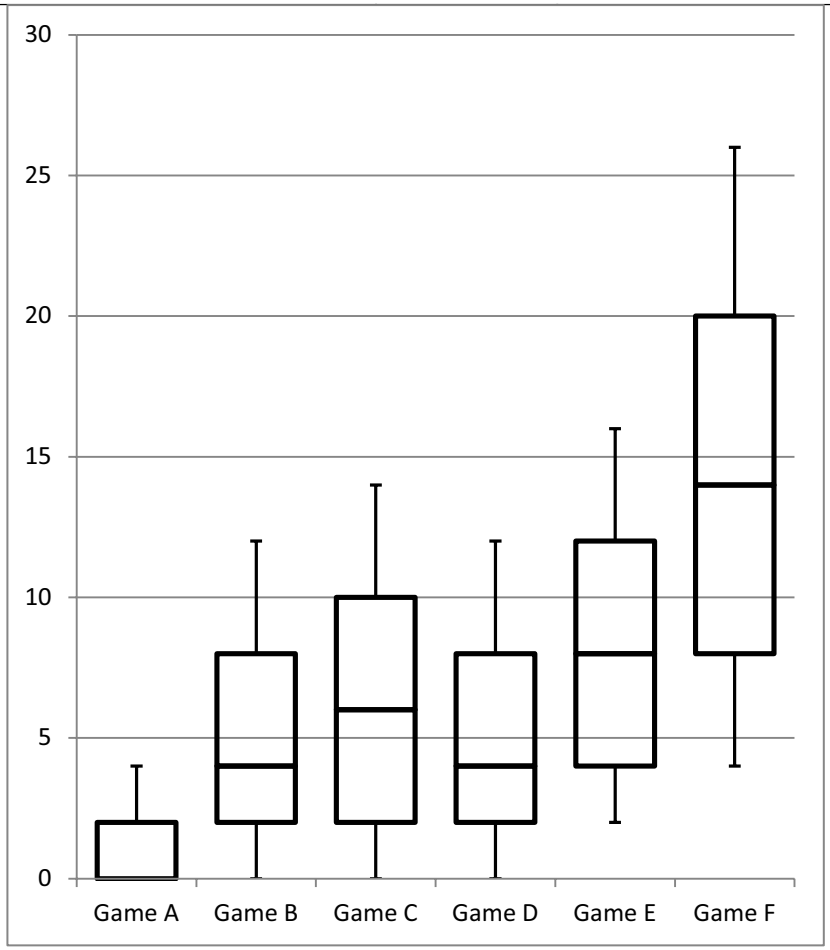

Fig. 10 Box plot for the results of six games used in human user survey

function is more entertaining as compared to the other evolved games

A one-tailed student test has been performed to check the above mentioned hypothesis. We check each of the games evolved using one metric (game *B*, game *C*, game *D*, and game *E*) and the randomly initialized game (game *A*) against the game evolved using the combined fitness function (Game *F*) to find the respective significant levels. The test is performed with *alpha* value of *0.05*, i.e., the significance level of *95%*. Table 4 shows the test results where game *F* is statistically tested against other five games. In each case, the value of *p* is much less than the *alpha* (*0.05*), so we are in a position to reject the null hypothesis (H_0). Thus, it is concluded that the game *F* is more entertaining. Fig. 10 shows the box plots of six games using the human user survey data of Table 2.

11. Conclusion and future work

The idea of evolution of game rules to produce new and entertaining games is very promising. In this work, we have presented a computational intelligence based approach for evolution of a platform type of computer game. In the process, we have presented some metrics for entertainment which are based on the duration of play, level of challenge, diversity, and usability of the play area. These metrics are combined in a fitness function to guide the search for evolving rules of the game using genetic algorithm. The results of our experiments are further verified against the human user entertainment value by conducting a human user survey and an experiment on the controller learning ability. We believe that there is great potential in further refining the entertainment metrics suggested in this paper. This work seeks to propose a solution by suggesting a 4-way fitness function and encourages research in the area of implementing automatic game content generation for platform games. Research in entertainment measurement involves fields ranging from artificial intelligence to psychology. Games can be significantly improved if we can quantitatively measure entertainment using an objective formula. Further work needs to be done for the validation of the results produced. Another direction can be to try different types of controllers for evaluating the chromosomes. Some other directions 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 at present, we combine the individual components of entertainment metrics linearly; an alternate would be to utilize multi objective genetic algorithm for this purpose.

References

1. C. K. Olson, L. A. Kutner, D. E. Warner, J. B. Almerigi, L. Baer, A. M. Nicholi, and E. V. Beresin, "Factors correlated with violent video game use by adolescent boys and girls", *Journal of Adolescent Health*, pp. 77-83, 2007.
2. Essential facts about the computer and video game industry: 2012 sales, demographic and usage data. *Entertainment Software Association*, 2012.
3. H. Iida, N. Takeshita, and J. Yoshimura, "A Metric for Entertainment of Board Games: Its Application for Evolution of Chess Variants", In Nakatsu, R., and Hoshino, J., eds., *Entertainment Computing: Technologies and Applications* (Proceedings of IWEC 2002), pages 65-72. Boston, MA: Kluwer Academic Publishers, 2003.
4. A. Cincotti and H. Iida, "Outcome Uncertainty And Interestedness In Game-Playing: A Case Study Using Synchronized Hex", *In New Mathematics and Natural Computation (NMNC)*, vol. 02, issue 02, pages 173-181, 2006.
5. S. Retalis, "Creating Adaptive e-Learning Board Games for School Settings Using the ELG Environment", *Journal of Universal Computer Science*: 2897-2908, 2008.
6. J. Togelius, R. D. Nardi, and S. M. Lucas,"Towards automatic personalised content creation for racing games", *in Proceedings of the IEEE Symposium on Computational Intelligence and Games*, Piscataway, NJ, 1-5 April, 2007.
7. G. N. Yannakakis , J. Hallam, "Towards Optimizing Entertainment In Computer Games", *Applied Artificial Intelligence*, v.21 n.10, p.933-971, November 2007.
8. M. Gallagher and A. Ryan, "Learning to Play Pac-Man: An Evolutionary Rule-based Approach," *in proceedings of IEEE Congress on Evolutionary Computation*, Canberra, Australia, 8-12 December, 2003.
9. Z. Halim, A. R. Baig and H. Mujtaba, "Evolutionary Search for Entertainment in Computer Games," *Intelligent Automation and Soft Computing*, Vol. 18, No. 1, pp. 33-47, January 2012.
10. Z. Halim, A. R. Baig and H. Mujtaba , "Measuring Entertainment And Automatic Generation Of Entertaining Games," *International Journal of Information Technology, Communications and Convergence*. Vol. 1, No. 1, pp. 92–107, January 2010.
11. G. Lankveld, P. Spronck, M. Rauterberg, "Difficulty Scaling through Incongruity", *in proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference*, Stanford, California, 22–24 October, 2008.
12. J. Togelius and J. Schmidhuber, "An Experiment in Automatic Game Design", *in Proceedings of IEEE*

Computational Intelligence and Games, Perth, Australia, 15 - 18 December, 2008.
13. J.Schmidhuber, ”Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts”, *Connection Science*, vol. 18, pp.173–187, 2006.
14. J. Togelius, S. Karakovskiy, J. Koutnik and J. Schmidhuber, "Super Mario Evolution", *in proceedings of IEEE Symposium on Computational Intelligence and Games*, Milano, Italy, 7-10 September, 2009.
15. N. Lazzaro, “Why We Play Games: Four Keys to More Emotion Without Story”, *in proceedings of Game Developers Conference*, San Jose, California, 8th March, 2004.
16. K.Compton and M. Mateas, “Procedural Level Design for Platform Games”, *in proceedings of 2nd Artificial Intelligence and Interactive Digital Entertainment Conference*, Stanford, 20-23 June, California, 2006.
17. G. N. Yannakakis, and J. Hallam, "Entertainment Modeling through Physiology in Physical Play," *International Journal of Human-Computer Studies*, vol. 66, issue 10, pp. 741-755, October 2008.
18. J. Kushida, I. Nakaoka, K. Kamei and Y. Hoshino, “Application of Co-Evolutionary System for Strategy Developments of Teams in the Same Generation to Team Match-Up Games”, *International Journal of Innovative Computing, Information and Control*, Vol. 5, No. 11(A), pp. 3667-3676 , 2009.
19. J. Kushida, I. Nakaoka, K. Kamei, N. Taniguchi and Y. Hoshino, “A Coevolutionary System for Strategy Development in Poker Games”, *International Journal of Innovative Computing, Information and Control*, Vol. 4, No. 12, pp. 3259-3272, 2008.
20. W. Baghdadi, F.S. Eddin, R. Al-Omari, Z. Alhalawani, M. Shaker, N. Shaker, "A Procedural Method for Automatic Generation of Spelunky Levels," *Applications of Evolutionary Computation*, Vol. 9028 pp. 305-317, 2015.
21. M. Shaker, N. Shaker, M. Abou-Zleikha, J. Togelius, "A Projection-Based Approach for Real-Time Assessment and Playability Check for Physics-Based Games," *Applications of Evolutionary Computation, Applications of Evolutionary Computation*, Vol. 9028 pp. 430-442, 2015.
22. L. Cardamone, P.L. Lanzi, D. Loiacono, "TrackGen: An interactive track generator for TORCS and Speed-Dreams," *Applied Soft Computing*, Vol. 28, pp. 550-558, 2015.
23. Z. Halim, A. R. Baig, K. Zafar, "Evolutionary Search in the Space of Rules for Creation of New Two-Player Board Games," *International Journal on Artificial Intelligence Tools*, Vol. 23, No. 2, April 2014.
24. H. Mujtaba, A. R. Baig, Z. Halim and A. Iqbal , “Self-Motivated and Task-Oriented, Multi-Dimensional Learning in a Dynamic and Uncertain Environment without Human intervention," *International Journal of Innovative Computing, Information and Control,* Vol.7, No.4, pp. 1603-1620, April 2011.
25. Z. Halim, A. R. Baig and S. Khan, "Modular Indoor Games: A Hybrid of Video and Outdoor Games", *International Conference on Computational Intelligence and Software Engineering (CiSE)*, Wuhan, China, December 11-13, 2009.
26. T.S. Nielsen, G.A.B. Barros, J. Togelius, M.J. Nelson, "General Video Game Evaluation Using Relative Algorithm Performance Profiles," *Applications of Evolutionary Computation, Applications of Evolutionary Computation, Applications of Evolutionary Computation*, Vol. 9028 pp. 369-380, 2015.
27. M. Csikszentmihalyi and I. Csikszentmihalyi, “Introduction to Part IV in Optimal Experience”, *Psychological Studies of Flow in Consciousness, Cambridge, UK*: Cambridge University Press. 1988.
28. T. W. Malone, “ What makes computer games fun?”, *ACM SIGSOC Bulletin*, Volume 13 , Issue 2-3 1982.
29. M. Federoff, “Heuristics and usability guidelines for the creation and evaluation of fun in video games”, Ph.D. Dissertation, Indiana University, Bloomington. Unpublished thesis, http://www.melissafederoff.com/thesis.html. 2002
30. M. Sharma and M. Mehta, M. Onta˜n´on, and S. Ram, ,”Player modeling evaluation for interactive fiction”, *in proceeding of third Artificial Intelligence for Interactive Digital Entertainment Conference*, California, USA, 20-23 June, 2007
31. D. H. Ackley and M. L. Littman,”Interactions between learning and evolution”, In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, pages 478–507, Reading, Sante Fe Institute Studies in the Sciences and Complexity, Addison-Wesley, MA, 1992.
32. R. Koster, “ A Theory of Fun for Game Design”, Paraglyph Press, 2005.
33. M. Rauterberg, ”About a Framework for Information and Information Processing of Learning Systems”, In Falkenberg, E.; Hesse, W.; and Olibve, A., eds., *Information System Concepts*, 54-69, IFIP Chapman & Hall. 1995
34. Essential Facts About The Computer And Video Game, Industry 2005 Sales, Demographics And Usage Data
35. T.W. Malone, "What makes things fun to learn? Heuristics for designing instructional computer games," *Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems*. ACM, 1980.
36. J. Togelius, G.N. Yannakakis, K.O. Stanley, C. Browne,"Search-based procedural content generation: A taxonomy and survey," *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 3, No. 3, pp. 172-186, 2011.