

Personalised Track Design in Car Racing Games

Theodosios Georgiou

Personal Robotics Laboratory

Department of Electrical and Electronic Engineering

Imperial College London

Email: theodosios.georgiou08@imperial.ac.uk

Yiannis Demiris

Personal Robotics Laboratory

Department of Electrical and Electronic Engineering

Imperial College London

Email: y.demiris@imperial.ac.uk

Abstract—Real-time adaptation of computer games’ content to the users’ skills and abilities can enhance the player’s engagement and immersion. Understanding of the user’s potential while playing is of high importance in order to allow the successful procedural generation of user-tailored content. We investigate how player models can be created in car racing games. Our user model uses a combination of data from unobtrusive sensors, while the user is playing a car racing simulator. It extracts features through machine learning techniques, which are then used to comprehend the user’s gameplay, by utilising the educational theoretical frameworks of the Concept of Flow and Zone of Proximal Development. The end result is to provide at a next stage a new track that fits to the user needs, which aids both the training of the driver and their engagement in the game. In order to validate that the system is designing personalised tracks, we associated the average performance from 41 users that played the game, with the difficulty factor of the generated track. In addition, the variation in paths of the implemented tracks between users provides a good indicator for the suitability of the system.

I. INTRODUCTION

“Future games are expected to have less manual and more user-generated or procedurally-generated content” [2]. This notion has the opportunity to elicit personalised and novel game content that can provide everlasting levels. In order to personalise you must first understand the player’s skills and abilities in a particular game. This involves knowing the game *Mechanics* – the components and rules of the game – that give rise to game *Dynamics* – how mechanics behave on user inputs – and fuses to game *Aesthetics* – user experiences invoked by the game (MDA framework) [3].

The main characteristics of an enjoyable game have been decoded by Malone [4] to be: challenge, fantasy and curiosity. In our research we are targeting both the training and engagement of the user in car racing games by stimulating the factors of game enjoyment through the user-adaptive procedural-generation of a track’s path. Pushing the car to the limits and handling tight turns at high speeds is what engages the users in that category of games. However, according to Steels [5], frustration and anger intensifies if the skills of the user are not sufficient enough to handle the difficulty of a given track.

We are proposing a framework where a combination of raw data from the game and sensors provided (e.g. eye tracking and head pose) are used to extract relevant features through machine learning techniques to implement a user model that is able to explain the current user’s performance during

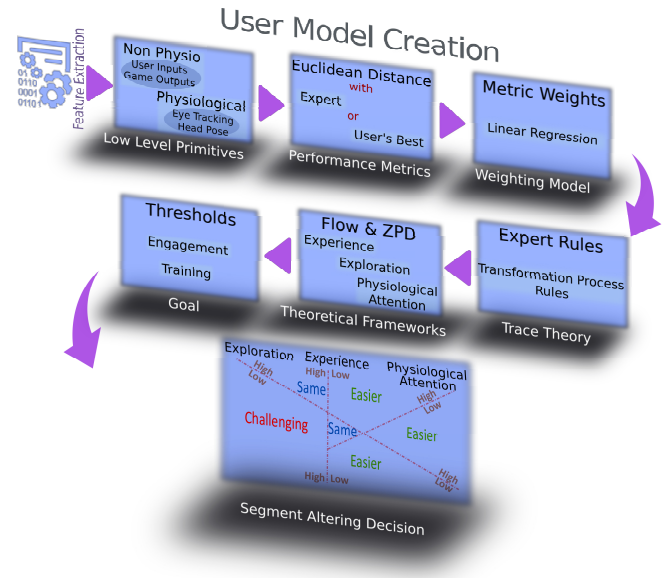


Fig. 1. Personalised user modelling approach for evaluating a driver in car racing game. Low level inputs are being converted to performance metrics where each metric defines the level of expertise of the user at a particular domain. For each domain a significance weight is assigned according to the individual and the task performed. Then game related rules transform metric-weight pairs to notions from the Concept of Flow [1] which are in turn exploited to evaluate the performance of the user and provide instructions on how the track’s path should be altered.

gameplay. Performance in games is a continuous function of the user’s skill and challenges, as well as the attention of the player. The aim of the framework is to: (a) monitor these properties, (b) update the user model, (c) provide decision adjustments for the alteration of the racing track according to the user, and (d) generate new tracks that suit the user profile. We propose a technique that modifies the game experience in real-time with the purpose of keeping the player’s satisfaction high and enhancing the learning process.

II. RELATED WORK

User-oriented track generation has been approached before by Togelius et al. [6], [7], [8]. Their evolutionary algorithm (*Cascading Elitism*) generated a number of different tracks either by changing the control points of a basic track segment or by constraining their angular position. Then a

neural-network based controller [7], that was trained on human driver behaviour, was testing if a generated track is challenging enough for a particular driver. Fitness metrics (e.g. varying challenge, fast driving regions) were used to evaluate the suitability of a new track for the controller. However, the research was focused on the methodology and creativity of the generated tracks instead of their evaluation with human drivers.

Taking the research to the next step, Loiacono et al. [9] derived an algorithm for generating new tracks in a car simulator (TORCS) using single and multi-objective genetic algorithms. By maximising the entropy of certain criteria (e.g. path curvature distributions along the track, achievable speeds distributions) and under the condition that the track has to be closed, their algorithm fills the path through particular “control” points that the road needs to pass through. Their initial aim was to provide tracks with an adequate amount of challenge and a large degree of diversity across their path. A further improvement, for embedding a human oriented decision to the algorithm, was proposed by Cardamone et al. [10] where the framework for advancing the algorithm to a next generation of tracks was also influenced by human assistance. Subjects voted for each generated track using scoring interfaces (5 Likert scale or boolean type) that were influencing the algorithm over the next generations of tracks. They showed that there was an improvement of user satisfaction in early generations. However, when the evolved tracks were tested by human subjects, they concluded that the tracks were only appealing to the players with some experience in racing games.

Apart for just being entertaining and stimulating, games can be used for training and educating as well. Those games are referred in literature as “Serious Games” [11]; a medium to enhance education in a more entertaining way. Based on the notion of teaching through games, Backlund et al. [12] conducted research on driver behaviour between racing, action, sport (RAS) gamers and non-gamers. People were categorised into two groups through a questionnaire, whereas their driving skills (attention, decision making, risk assessment, etc.) and attitude (respect speed limits, speed margins and fellow drivers) were rated by driving school instructors, using 7-point Likert scale. Their findings show a positive correlation between gaming and skill oriented aspects of driving. An important concluding statement suggested that games, and more specifically driving simulators, are able to provide positive effects on driving behaviour and user skill enhancement thus motivating further research.

III. SYSTEM OVERVIEW

A. The Simulator

In order to test the functionality of our user model and the performance of the track design algorithm, actual game-play data were collected from 41 users through high-end commercial racing car simulator game, called *rFactor 2*. The game provides an Application Program Interface (API) that allows the output of real-time data from the game environment

as well as user inputs. It also allows the introduction of new tracks into the game. Our custom made car simulator (see Fig. 2) is equipped with Vision Racer VR3 seat, Logitech G27 Force Feedback Steering Wheel and a combination of three monitors to enhance the user’s immersion to the game. Also, a Tobii EyeX eye tracker is installed in order to monitor the gaze of the user and 2 RGB-D cameras (Kinect) are used for capturing the player’s face, head pose, actions and the output from the monitors. Each user was asked to complete 20 laps on a particular track that we developed and answer a few questions before and after playing the game in order to assess their performance through the model.

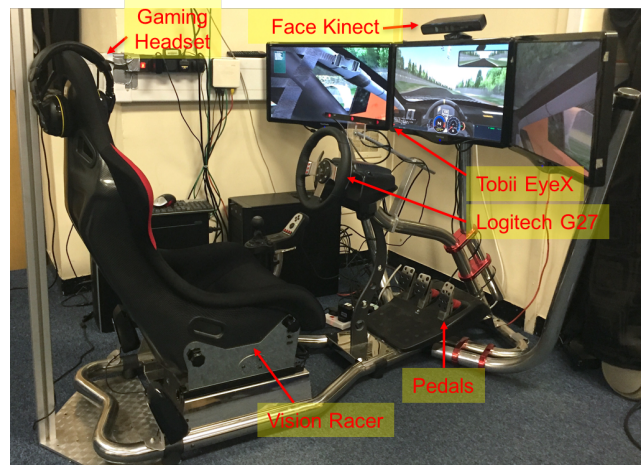


Fig. 2. Our custom car simulator setup consisting of Vision Racer Seat with a force feedback steering wheel, pedals and three monitors to enhance the user’s immersion to the game. Two RGB-D cameras as well as an eye tracker was installed to capture user’s gaze and pose positions.

B. The User Model

The aim of our user model is to capture the user’s current state through the available data and provide decisions for the future path of the track that suits the particular profile of the player. The user model structure, shown in Fig. 1, performs a series of analysis on the incoming user data to reach the decision state.

1) *Feature Extraction*: Features from user inputs, game outputs, eye tracker and head pose are extracted from the raw data provided through the game API and sensors and are grouped according to the location of the user in predefined paths of the selected track (*segments*). As seen in Fig. 3, segments are defined in such a way that they will consist only of a single path type (e.g straight, turn, chicane, etc.). The selected track is short ($\approx 1.93\text{Km}$) but still challenging track with various types of segments.

Table I lists the features collected, categorised by their sources. It is important to mention that head pose is established by passing the depth data from the RGB-D sensor, facing the user, through pre-trained Random Forests (RFs), that estimate the 3D coordinates of the nose tip and the angles of the head’s rotation [13]. Also, the clustering process to obtain the center

TABLE I
FEATURES OF THE LOWEST LEVEL OF THE USER MODEL

User Inputs	Game Outputs	Eye Tracker	Head Pose
1. Average Braking	6. No. of collisions	10. No. of Blinks/s	17. CN of HP
2. Average Throttle	7. Car XYZ Position	11. Screen Concentration	18. CC of HP
3. Average Steering	8. Position and Speed	12. CN of EG	19. CN of HP and VO
4. CN of User Inputs	9. (Segment Time)	13. CC of EG	20. CC of HP and VO
5. CC of User Inputs		14. CN of EG and VO	
		15. CC of EG and VO	
		16. Eye Fixations	

— CN: Cluster Number, CC: Cluster Centres, VO: Virtual Orientation, EG: Eye Gaze, HP: Head Pose

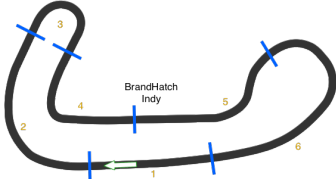


Fig. 3. Track used for the experiments split manually into segments

locations and cluster number of eye gaze and head poses features, is performed using the Affinity Propagation (AP) algorithm [14]. Features with Virtual Orientation (VO) are those whose raw data are paired with the respective sequences of virtual world angles of the car in the game.

2) *Performance Metrics*: Every time a user completes a segment, the features (from Table I) are calculated and compared for proximity to user's previous "player_best" features to form the Performance Metrics. The "player_best" are determined as these with the fastest segment time feature. The "player_best" set is re-initialised every time the user does a better segment time and all of the previous metrics of that segment are re-computed. In order to reduce the outliers, each of the metrics, in the feature vector is converted to a percentage value using the exponential function shown in (1). The X is the highest value a metric can obtain. It is determined when a "player_best" feature is compared with itself to obtain a metric. The Constant, C , for each metric was determined empirically off-line through all the data collected so that their median value lies around 50%.

$$\%Metric = 100 * \exp\left(-\frac{(Metric - X)}{C}\right) \quad (1)$$

3) *Weighting Model*: Metrics are classified according to two groups: the *Physio* group, which includes metrics associated with user's physiological data, such as those obtained from eye tracker and head pose and the *Non-Physio* group, which consists of metrics obtained from user inputs and game outputs. The user model is based on how much each of the metric created is correlated with the performance of the user on a particular segment. Good performance means low segment time. As we will explain later, the user model is providing decisions according to three notions: the *Experience*, the *Challenge* and the *Attention* of the user to the game. For each of these notions there is a principle that uses a subset of

the available metrics. The subsets are: *All* – which contains all metrics – *Physio* (p) and *Non-Physio* (np). For each metric in each subset we determine a weight according to the proportionality shown in (2).

$$t_s^l \propto \sum_{k=1}^m z W_k^s M_k^{sl} \quad (2)$$

where:

- t_s^l is the time performance metric of a segment s and user lap l (No. 9 in Table I)
- $z W_k^s$ is the weight for a particular segment s of a metric k in the subset z .
- m is the number of metrics defined in each of their group: All, Non-Physio, Physio.
- M_k^{sl} is the Performance Metric k of segment s and lap l

In order to find the weights $z W_k^s$ we are treating (2) as a linear regression (LR) model. This is solved using the least squares (LS) problem method by minimising the sum of the squared error ($\min \|Ax - b\|_2^2$) where in our case A represents a matrix of our metrics observations and b is their corresponding time metrics. The algorithm is forced to provide either positive or zero valued coefficients which are then normalised to find the weights $z W_k^s$. The reasoning for the positive bounds is because metrics were designed to have positive correlations to time and also the conversion of the metrics to percentages was one sided with 100% being the user's "player_best". Weights are re-evaluated every-time a new set of features are created for a particular segment.

4) *Transformation Rules*: Following the theoretical frameworks of behavioural analysis like the Concept of Flow [1], [15], [5], the Zone of Proximal Development [16] and the Trace-Based System theory [17], [18] we further analyse the subsets of metrics using game-specific assumptions into certain rules and principles in order to form three classes that make up the high level in the user model: Experience, Exploration and Physiological Attention.

a) *Experience (E)*: (or the skills of the user) is determined by the proximity of the user's metrics to the "player_best" ones. Since the metrics are expressed as percentages, the higher the value the better the user is performing on that particular metric. The significance of this metric to the user's skill is determined by the weight calculated in the *All* subset. Also, skill cannot be determined by a single

group of values. It has to be an overall value of several trials. Therefore, it is calculated by the weighted sum of the mean of each performance metric over a certain number of laps for a particular segment, as shown by (3). In this paper we considered 10 laps since they were enough to show the skills and avoid the overtraining of the particular track.

$$Experience_n^{l,s} = \sum_{k=1}^m all W_k^s * \frac{\sum_{i=l+(1-n)}^l M_k^{i,s}}{n}, \quad (3)$$

$$l > n, n > 0, s \in S$$

b) *Exploration (C)*: In the racing game, exploration can be explained when the user tries different approaches and techniques to complete a segment. This can be either different racing lines, unusual eye fixations, new input averages, etc. Through the metrics we define the Experience for each segment as the weighted sum of the mean of the consecutive “jump” ($all J_k^s$) of each of the performance metrics in the *All* subset. If the difference between two consecutive metrics passes a fixed percentage value of the current experience then the value is positive, otherwise it is negative. This is shown by (4).

$$Exploration_n^{l,s} = \sum_{k=1}^m all W_k^s * \left((|M_k^{l,s} - M_k^{l-1,s}|) - (all J_k^s * \frac{\sum_{i=l+(1-n)}^l M_k^{i,s}}{n}) \right),$$

$$1 > J_k > 0, l > n, n > 0, l > 1, s \in S \quad (4)$$

c) *Physiological Attention (PA)*: is designed to keep record of the continuous attention of the user along consecutive segments. This is calculated by first evaluating the Experience (*E*) of the user only from non-physiological (*np* subset) data through consecutive segment data. If the value is above a threshold (^{np}T) then the assumption is that the user’s attention is high, since (s)he performs well, and the value is kept. Otherwise, we calculate the Exploration (*C*) value only from the physiological data (*p* subset) and use that for physiological attention. This is expressed by (5).

$$I = f(l, s),$$

$$^{np}E_n^{l,s} = \sum_{k=1}^{^{np}m} ^{np}W_k^s * \frac{\sum_{i=I+(1-n)}^I M_k^{i,s}}{n},$$

$$^pC_n^{l,s} = \sum_{k=1}^{^pm} ^pW_k^s * \left((|M_k^{I,s} - M_k^{I-1,s}|) - (^pJ_k^s * \frac{\sum_{i=I+(1-n)}^I M_k^{i,s}}{n}) \right), \quad (5)$$

$$Attention_n^{l,s} = \begin{cases} ^{np}E_n^{l,s} - ^{np}T & \text{if } ^{np}E_n^{l,s} \geq ^{np}T \\ ^pC_n^{l,s} & \text{if } ^{np}E_n^{l,s} < ^{np}T \end{cases}$$

$$1 > ^pJ_k^s > 0, l > n, n > 0, l > 1, s \in S$$

where: *I* is the index number representing segment *s* in lap *l*. Function *f* transforms *s* and *l* to *I* so that *I* – 1 defines

the previous consecutive segment of index *I*. *n* defines the number of laps from which the value is calculated.

5) *Concept of Flow and ZPD*: By defining these three notions we are able to follow the Theory of Flow [1] which is defined to be the feeling of being completely immersed and engaged in an activity and also experiencing high levels of enjoyment and fulfilment [15]. Steels [5] states that for a person to remain engaged with a task and in the “flow” there has to be balance between the level of challenge (Exploration) and user’s skills (Experience). However, when the Attention of the user is low these values are more sensitive to each other. By using thresholds, the user model provides instructions for each segment (keep same, easier, more challenging) to the track design algorithm (described in Section III-C) so as to provide new relevant segments customised to the user.

The main idea of the model is also to utilise the theoretical framework of Zone of Proximal Development (ZPD) [16] which defines the difference between what a person can achieve with (Potential Development (PD)) or without help (Actual Developmental Level (ADL)). Therefore, if the user is becoming skilled enough (high Experience) on particular segments, then the segment should become more challenging in order to increase the user’s PD and encourage more training. On the other hand, if (s)he cannot cope with the current path then it should become easier since the user’s ADL is not enough for the particular segment. The bottom part of Fig. 1 (Segment Altering Decision) summarises the output of the user model according to predefined thresholds for each notion.

C. Track Design Algorithm

The proposed idea of this paper is to alter the segments of a track according to the model of the user. Therefore, we start from a track that already exists from which we evolve and generate new ones by changing its segments. For each segment of the track, as shown, via a single segment, in Fig. 4, we keep 3 kinds of data:

- 1) Detailed (*XYZ*) point representation of the center path of the segment.
- 2) The orthogonal distance of each center point to the edges of the road so that we know where the right and left sides are located.
- 3) The point representation of the optimal path on that segment paired with the car speed (if the data are available from an expert).

In order to be able to compare the paths created by the user to the optimal or center paths of the segment, we parameterise each path by fitting its points into an *n*th order Bezier Curve. Through experimentation we found that an 9th order Bezier is accurate and efficient enough to describe a segment path. A sequence of Bezier curves (b-splines) were also used by Togelius et al. [6], [7], [8] to represent the path of the track.

A Bezier curve equation is described by (6). The number of control points *P_i* describing the curve depend on the order of the degree chosen and in our algorithm their locations are determined by finding the best fit through the data points using the least squares (LS) method. The objective of the LS method

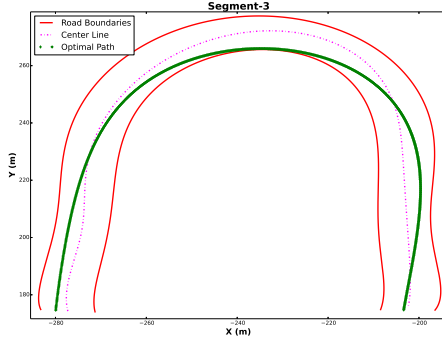


Fig. 4. Red lines define the boundary of the road. Dotted line is the center path and the green line defines the optimal path created by an expert.

is to minimise the sum of the squared errors (7). Each error (8) is defined as the difference between a path's data point (d_k) on one of the axes and the value obtained by the best fit equation of the curve (6). Calculating the partial derivatives of (7) and setting the result to zero as shown by (9) we end up with n parameters (P_i) and n gradient equations that can be arranged in a matrix form (10) and solve by using matrix inversion (11). By having the Bezier equation of a path enables us to extract a fixed number of XYZ points (by adjusting t_k divisions) and to compare it with other paths. In order to increase efficiency and also make sure that each equation starts and ends on the location we need them to, P_0 and P_n are fixed to the starting point and the ending point of the relative segment.

$$B(t_k) = \sum_{i=0}^n \binom{n}{i} (1-t_k)^{n-i} t_k^i P_i, \quad t_k \in [0, 1] \quad (6)$$

$$M = \sum_{k=0}^n e_k^2 \quad (7)$$

$$e_k = d_k - B(t_k) \quad (8)$$

$$\frac{\partial M}{\partial P_i} = -2 \sum_{k=0}^n e_k \frac{\partial e_k}{\partial P_i} = 0 \quad (9)$$

$$AX = C \quad (10)$$

$$X = A^{-1}C, \quad X^T = (P_0 \quad \dots \quad P_n) \quad (11)$$

The track design algorithm keeps track of the user's paths along each segment in the form of a Bezier equation and stays on hold until a "segment change decision" is received from the model. On the instruction to keep the *Same* segment, then no action is performed. When an *Easier* segment is needed then a mean path from previous user paths is calculated. This mean path serves as the optimal path to the center of the new segment that is created. There is one assumption here that the user will find a path easier if the optimal route (s)he has to perform is already experienced in previous trials. An example of a user adapted easy segment is shown in Fig. 5.

If there is a need of a more *Challenging* segment then a different procedure is performed. Here we assume that a path can be converted to a more difficult one by either (a)

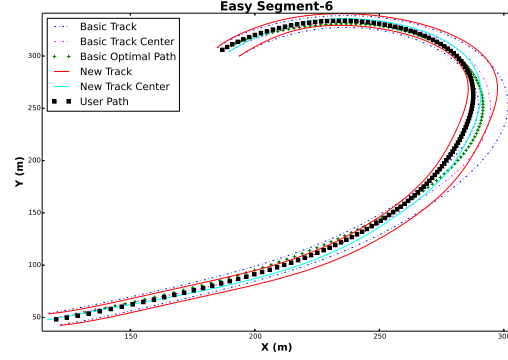


Fig. 5. The new center has been shifted in such a way that the average of the paths of the user is the optimal path of the new segment.

increasing the angle of its curve, (b) increasing the number of turns or (c) both. We are using a back-tracking algorithm to perform path planning between two points by setting a number of constraints. The algorithm proceeds from a starting point towards an end point by randomly checking for valid points along the circumference of a fixed radius. The constraints are set such as:

- There are no loops in the path, so that the path can be constructed in the game.
- The lines created by 3 points should not have an angle less than 145° , so that to keep the path smooth.
- A new point should be at a fixed distance (d) from any orthogonal point on the current path, so that to avoid any intersections between other segments (the value d depends on the basic track selected).
- The new path should be at least 80% different from any of the previous paths of the user.
- The new path should start and end where the previous path was, in order to be able to plug and play the new created segment.

A challenging example is shown in Fig. 6. When a new path is created then the points are fit into a Bezier curve with the additional constraint that the end points gradient should stay the same as the path they are replacing. This refines the points and preserves the seamless link between the segments.

D. Graphics Generation

In order to be able to generate a track model that can be imported into the game, we need to create a track in a more specialised environment that is also supported by the game. We are using a computer graphics program called *Autodesk 3ds Max 2012* for which the particular game has a graphics model export plugin. The basic track was manually designed in the 3D software in a way that every object in the scene is virtually attached to the center path line of the track. The advantage of this method is that by altering the points of the center line to a new location then all the graphics objects are modified as well to adapt to the new change. Therefore, when creating a new segment we only need to alter the points specified in the

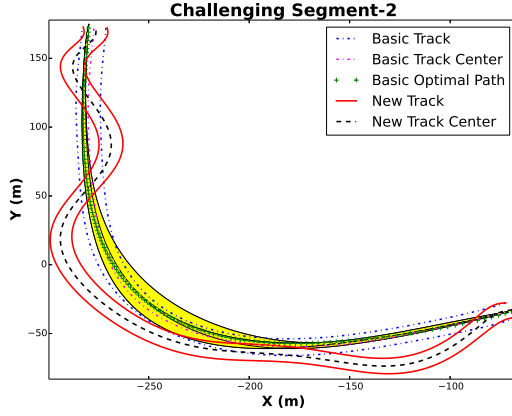


Fig. 6. Red boundaries show the path constructed by the backtracking algorithm. The previous segment path is displayed in dotted blue lines and the yellow shaded area defines the variance of the paths covered by the particular user in her previous attempts.

software for the particular segment. The basic track's graphic model and its rendering by the game used in our experiments are shown in Fig. 7.

One of the possible ways to describe a line between two points in the software is by using sequences of *cubic bezier splines*. This is using two points for defining the starting and ending location of the curve and two control points that determine the path curvature between the two points. Consecutive lines share location points (named as *knots*). Therefore, the track design algorithm has to convert the n^{th} Bezier spline segment of a new path into multiple cubic bezier splines under two conditions:

- It should have the same starting and ending point to the basic track segment, so that there is a smooth continuation between the segments.
- The total number of knots of each segment should be the same as the basic track's so that their indexing in the software is always the same. This restriction is not compulsory but if the number of knots change then the index configurations of all segments should update as well. Also, the knots are quite dense so there is no problem expanding the length of the path.

Evaluation of user model, decision taking and track design are all happening online. The track design algorithm is communicating with the graphics software through a custom TCP client-server program. The *client* is controlled by the track design algorithm and is responsible for sending (a) the appropriate segment identification and also (b) the software ready locations of knots and control points. The *server* is responsible for (a) communicating with the graphics software and (b) calling the software's scripts for altering the knots and their control points, (c) creating the graphics model and (d) sending the new track model files to the game.

IV. RESULTS

The user model outputs three instructions on how each segment should change in the evolved track (easier, same, challenging). Our basic track consists of six segments from which five are supposed to change through the framework (start line – segment 1 – is omitted due to game related constraints). Therefore, this gives $3^5 - 1 = 242$ new permutations, whereas each one is unique to the user as well as the randomness of the backtracking algorithm. In order to validate for the diversity and personalisation of our framework, we used the data collected from 41 users to run offline track generation at the point where the users performed 10 out of 20 laps. The generated tracks of 12 randomly selected users are shown in Fig. 9.

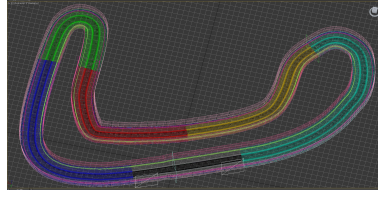
The users' profile was broad with ages between 19-35 and various expertise in racing games (31% non-gamers, 69% gamers) and driving (0–18 years). Because the purpose of the framework is set to provide training to the user, it is not supposed to deviate greatly from the basic track, since every segment starts and ends at the same location. From the shape of each track we can see that even with similar change instructions (defined by the change id: C - Challenging, S - Same, E - Easier) the segments are not the same, except if no change is performed. In order to test for user profile customisation we correlated the overall difficulty of the track generated to the average time taken for the user to complete the 10 laps. Difficulty was determined by assigning a score to the instructions (e.g. $C = 2$, $S = 1$, $E = 0$) and summing them over all segments. In Fig. 9, difficulty is shown by the number in the brackets. Therefore, the basic track has a difficulty score of 5 since $ChangeID = SSSSS$, whereas the range varies between 0 – 10.

The Spearman correlation test between *average time* and *track difficulty* from all user data gave a statistically significant value of -0.82 ($p < 0.01$). This shows that there is a strong negative correlation between the average lap time and the track difficulty value generated by the model. It is important to mention that the user model is not directly using the time taken for the user to complete each segment in order calculate the performance of the user.

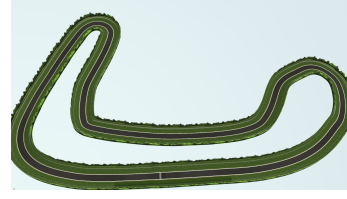
By assigning each user to a group using their average time over the 10 laps, according to Table II, we were able to utilise *Kruskal-Wallis* statistical test to investigate if the track difficulty assignment to the different groups is coming from the same distribution (random assignment). The test returned $\chi^2 = 29.12$, $df = 3$, $p - value < 0.01$ which gives us enough evidence to reject the null hypothesis. The time thresholds (specified in Table II) for grouping the users were set according to the largest gaps that were formed between subjects' average time.

A. Discussion

We can notice a clear distinction between Advanced-Intermediates groups and Average-Beginners groups (Fig. 8). The former typically gets difficulty values above 6 whereas the latter generates values around and below 6. Therefore, the



(a) Wireframe from Graphics software



(b) Game Rendered Track

Fig. 7. The basic track's path is similar to a real racing track called "BrandHatch Indy" which is located in London (UK). It is considered to be a short($\approx 1.93\text{Km}$) but still challenging track with various types of segments.

general idea that more experienced users tend to get more challenging tracks than inexperienced ones is inferred by the framework.

The user model is designed to take account of the real-time attention of the user, in addition to the user's skills and challenges. Consequently, if the user is self-motivated to learn, boost their skills and achieve better lap times during longer training sessions (>10 laps), the user model instructs most of the segments to stay the same and as a result track difficulty is closer to 5 (baseline). On the contrary, if the user lacks attention then, depending on their skill level, the model tries to accommodate either more challenging or easier segments. This increases the engagement of the user in the game and avoids the build up of boredom (when high experienced) or frustration (when low experienced). Every person is unique in their way of learning and training, this reason explains the variability between each group in Fig. 8.

TABLE II
AVERAGE TIME VERSUS EXPERIENCE ASSIGNMENT FROM DATA TAKEN FROM THE BASIC TRACK.

Experience	Average Time(%)	Players
1. Advanced	$t \leq 60$	5
2. Intermediate	$60 < t \leq 70$	14
3. Average	$70 < t \leq 83$	15
4. Beginner	$t > 83$	7

The proposed framework relates to the general Experience-Driven Procedural Content Generation (EDPCG) framework proposed by [19]. The low level of the model follows a *hybrid* player experience modelling (PEM). It extracts information from unobtrusive *objective* PEM combined with *gameplay-based* inputs and outputs of the racing game. The extraction of data and weights follows a *model-free* approach whereas the higher levels follow a *model-based* using theoretical frameworks. Through this paper, the evaluation of the game content is performed using a *data-driven direct* functions, as the expertise of the user is correlated with the difficulty of the implemented track. In an ongoing future work our aim is to perform evaluation via *interactive implicit* and *explicit* functions by assessing the overall training of the user and by analysing the user feedback. Content representation, generation and optimisation is the main focus of this paper and according to the EDPCG framework it is a form of a

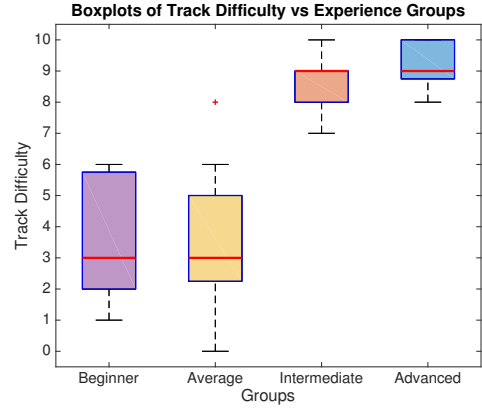


Fig. 8. Box plots of the track difficulty against user's experience group. *Kruskal-Wallis* test shows statistically significant results: $\chi^2 = 29.12$, $df = 3$, $p\text{-value} < 0.01$ and Spearman test shows a strong negative correlation: -0.82 ($p < 0.01$)

search based approach.

V. CONCLUSION

We have implemented a framework for altering the path of the track online according to the skills, challenges and attention of the user, using physiological and non-physiological features through machine learning techniques, game-related rules and theoretical frameworks. To evolve the track we are employing the past history of the user, the decisions of the framework and the functionality of the Bezier curves with the aim to train and also keep the user engaged.

In our results we showed that the framework follows the general trend regarding the skills of the user and provides diversity between and among the subjects. To the extent of our knowledge, this is the first attempt to evolve the track of a 3D-game in situ, using real-time input from the user while is playing the game. User adaptive generation of tracks is a very interesting domain which can alter the way racing games are being designed and increase the market of consumers due to their flexible and unlimited content. In the future we would like to extensively test the evolved tracks with human subjects.

ACKNOWLEDGMENT

The authors would like to thank Loizos Shianios for the help provided on designing the track in the specialised software.

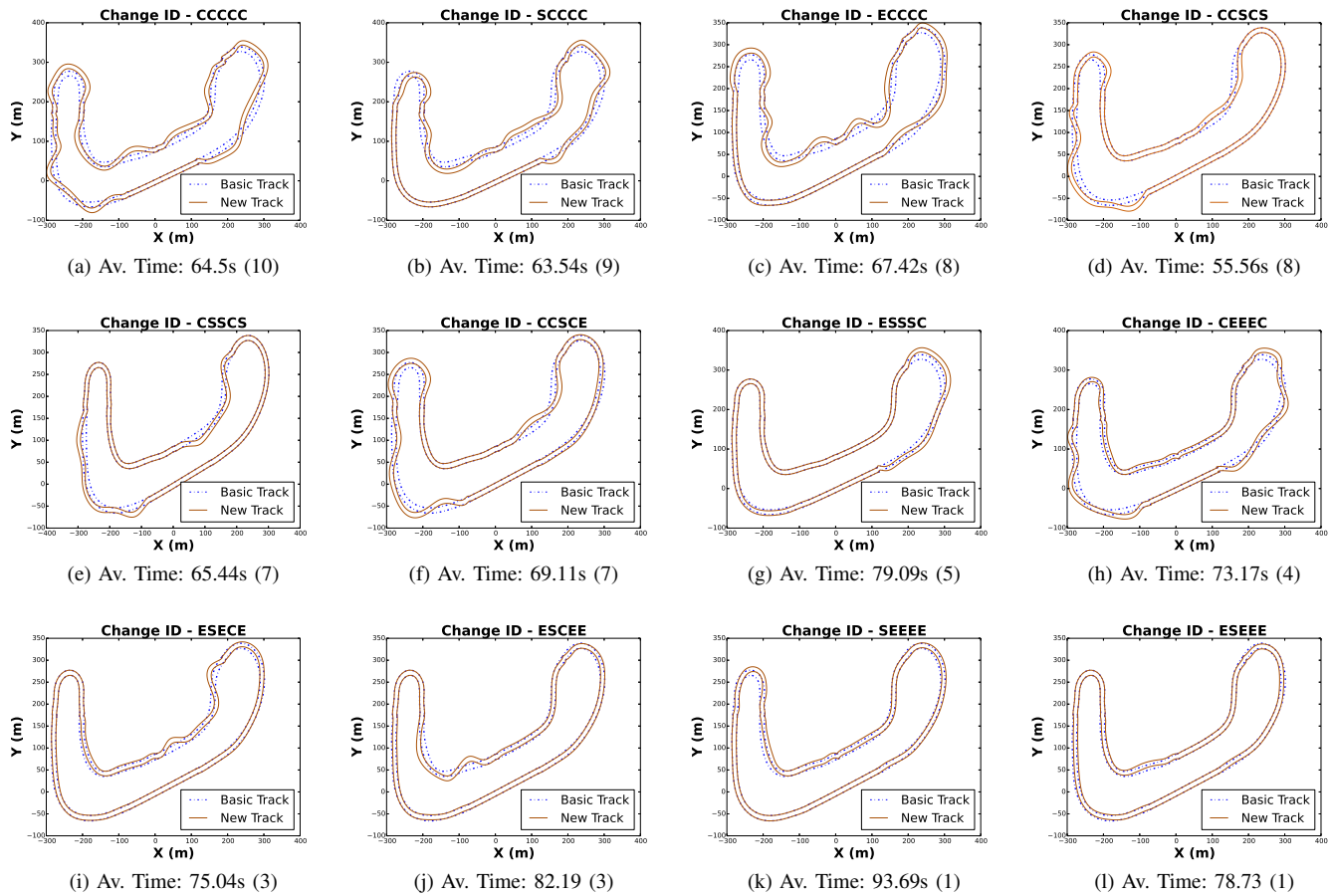


Fig. 9. Tracks generated through different users after the model collected data over 10 laps. The number in the brackets indicates the difficulty level of the new track according to the changes performed on the basic one, whereas the “Average Time” is an indication of the skill of the particular user (Difficulty range varies between 0 – 10)

REFERENCES

- [1] J. Nakamura and M. Csikszentmihalyi, “The concept of flow,” in *Flow and the foundations of positive psychology*. Springer, 2014, pp. 239–263.
- [2] G. N. Yannakakis, P. Spronck, D. Loiacono, and E. André, “Player Modeling,” in *Artificial and Computational Intelligence in Games*. Schloss Dagstuhl Series, 2013, vol. 6, pp. 45–59.
- [3] R. Hunnicke, M. LeBlanc, and R. Zubeck, “MDA: A Formal Approach to Game Design and Game Research,” in *Proceedings of the AAAI Workshop on Challenges in Game AI*, vol. 4, 2004, pp. 1–4.
- [4] T. W. Malone, “What makes things fun to learn? Heuristics for designing instructional computer games,” in *Proceedings of the 3rd ACM SIGSMALL symposium*, 1980, pp. 162–169.
- [5] L. Steels, “The Architecture of Flow,” in *A Learning Zone of One’s Own*. IOS Press, 2004, pp. 135–150.
- [6] J. Togelius, R. De Nardi, and S. M. Lucas, “Making Racing Fun Through Player Modeling and Track Evolution,” in *Proceedings of the SAB Workshop on Adaptive Approaches to Optimizing Player Satisfaction*, 2006, pp. 61–70.
- [7] J. Togelius and S. M. Lucas, “Evolving robust and specialized car racing skills,” in *IEEE International Conference on Evolutionary Computation*, 2006, pp. 1187–1194.
- [8] J. Togelius, R. De Nardi, and S. M. Lucas, “Towards automatic personalised content creation for racing games,” in *IEEE Symposium on Computational Intelligence and Games*, 2007, pp. 252–259.
- [9] D. Loiacono, L. Cardamone, and P. L. Lanzi, “Automatic track generation for high-end racing games using evolutionary computation,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 245–259, 2011.
- [10] L. Cardamone, D. Loiacono, and P. L. Lanzi, “Interactive evolution for the procedural generation of tracks in a high-end racing game,” in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM, 2011, pp. 395–402.
- [11] M. Zyda, “From visual simulation to virtual reality to games,” *IEEE Computer*, vol. 38, no. 9, pp. 25–32, 2005.
- [12] P. Backlund, H. Engström, and M. Johannesson, “Computer gaming and driving education,” in *Proceedings of the ICCE 2006 workshop Pedagogical Design of Educational Games*, 2006, pp. 9–16.
- [13] G. Fanelli, J. Gall, and L. Van Gool, “Real time head pose estimation with random regression forests,” in *Computer Vision and Pattern Recognition*, 2011, pp. 617–624.
- [14] B. J. Frey and D. Dueck, “Clustering by passing messages between data points,” *Science*, vol. 315, no. 5814, pp. 972–976, 2007.
- [15] M. Csikszentmihalyi, *Flow: The psychology of optimal experience*. New York: Harper and Row, 1990.
- [16] L. Vygotsky, “Interaction between learning and development,” *Readings on the development of children*, vol. 23, no. 3, pp. 34–41, 1978.
- [17] L. S. Settouti, Y. Prie, J.-C. Marty, and A. Mille, “A trace-based system for technology-enhanced learning systems personalisation,” in *Ninth IEEE International Conference on Advanced Learning Technologies*, 2009, pp. 93–97.
- [18] P. Bouvier, K. Sehaba, and E. Lavoué, “A trace-based approach to identifying users engagement and qualifying their engaged-behaviours in interactive systems: application to a social game,” *User Modeling and User-Adapted Interaction*, vol. 24, no. 5, pp. 413–451, 2014.
- [19] G. N. Yannakakis and J. Togelius, “Experience-driven procedural content generation,” *IEEE Transactions on Affective Computing*, vol. 2, no. 3, pp. 147–161, 2011.