Closing the Affective Loop via Experience-Driven Reinforcement Learning Designers

Matthew Barthet[1], Diogo Branco[2], Roberto Gallotta[1], Ahmed Khalifa[1], Georgios N. Yannakakis[1]
[1]Institute of Digital Games, University of Malta, Msida, Malta
Email: {matthew.barthet, roberto.gallotta, ahmed.khalifa, georgios.yannakakis}@um.edu.mt
[2]Faculty of Exact Sciences and Engineering, University of Madeira, Madeira, Portugal
Email: diogo.branco@arditi.pt

***Abstract*—Autonomously tailoring content to a set of predetermined affective patterns has long been considered the holy grail of affect-aware human-computer interaction at large. The experience-driven procedural content generation framework realises this vision by searching for content that elicits a certain experience pattern to a user. In this paper, we propose a novel reinforcement learning (RL) framework for generating affect-tailored content, and we test it in the domain of racing games. Specifically, the experience-driven RL (EDRL) framework is given a target arousal trace, and it then generates a racetrack that elicits the desired affective responses for a particular type of player. EDRL leverages a reward function that assesses the affective pattern of any generated racetrack from a corpus of arousal traces. Our findings suggest that EDRL can accurately generate affect-driven racing game levels according to a designer's style and outperforms search-based methods for personalised content generation. The method is not only directly applicable to game content generation tasks but also employable broadly to any domain that uses content for affective adaptation.**

***Index Terms*—affective computing, procedural content generation, reinforcement learning**

I. Introduction

One of the most challenging tasks within affective computing (AC) [1] is to effectively leverage affect models to autonomously generate new contexts that are, in turn, capable of eliciting a desired emotional response [2]. In other words, the challenge for AC is to successfully enable an affect-aware closed-loop adaptive system, largely known as the *affective loop* [3], [4]. What makes the design of such affect-aware adaptive interactions very difficult is the unpredictability and subjectivity of human users, both behaviourally and emotively.

Motivated by this challenge, we introduce a novel method for autonomously generating content that, when experienced, elicits a desired sequence of emotional responses to a particular user. To test the proposed framework, we focus on the domain of games, as they offer rich forms of human-computer interaction and have proven to be an ideal test bed for AC research in the past [4], [5]. In particular, we leverage human arousal demonstrations from over 100 annotators of a real-time 3D rally driving game from the Arousal video Game AnnotatIoN (AGAIN) corpus [6] to generate racetracks that elicit a desired arousal trace for a particular player type. To accomplish this, we build upon the experience-driven procedural content generation (PCG) via reinforcement learning (EDRL) framework [7] by using a simulation-based approach to reward the racetracks EDRL generates, as visualised in Fig. 1. We compare the performance of the EDRL approach against an experience-driven PCG [7] method that uses genetic search for creating racetracks. Both methods employ the K-Nearest Neighbours (KNN) algorithm [8] to generate arousal traces from provided game states encountered during simulations. The result is a generative AI framework that can adapt its generated outcome to multiple player and annotator types, and can be refined over time with more data.

We assess the efficacy of our methods by testing them across three different clusters of human annotators and three indicative target arousal patterns. We both quantitatively com-

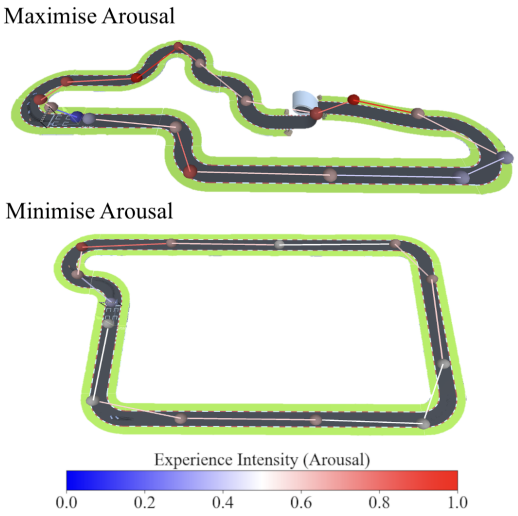

Fig. 1. Two examples of maximally and minimally arousing tracks generated by EDRL for the *Solid Rally* racing game. Top image: EDRL offers content that constantly elicits higher levels of arousal, such as loops and sequences of alternating turns. Bottom image: EDRL generates simple and straight tracks, leaving limited room for highly arousing gameplay. Blue overlay elements denote regions with a strong decrease in arousal, while red denotes regions of a strong increase in arousal, and white denotes neutral arousal changes.

This project has received funding from the Malta Council for Science and Technology through the SINO-MALTA Fund 2022, Project OPtiMaL. Diogo Branco was supported by Fundação para a Ciência e Tecnologia (FCT) under the PhD grant 2021.05646.BD and by the Project "Consortium Advanced Computing - HPC, HPDA, AI & HPV" (2021-1-PT01-KA131-HED-000008876) under the Erasmus+ Program, Education and Training 2021 - Higher Education - Action KA131 - Mobility for Learning Purposes.

979-8-3315-1643-7/24/$31.00

pare the ability of the methods to match the target arousal traces, and we conduct an in-depth analysis of the content generated and their associated play traces. Our findings suggest that the introduced EDRL method for continuous affect-driven generation is more efficient and robust than any other method tested. Moreover, findings suggest that certain affective patterns are harder to elicit—via level generation—than others for particular player types. Unsurprisingly, it is easier to generate maximally arousing tracks than minimally arousing tracks for players whose annotated arousal levels are kept low in this game. Inversely, players with strong increases and high arousal values were more difficult to satisfy emotionally when EDRL is tasked to minimize their arousal changes.

II. Related Work

In this section, we first cover related studies at the intersection of AC and games in general (Section II-A) and then specifically AC studies as applied to procedural content generation (Section II-B).

A. Affective Computing in Games

Applications of AC span a vast array of contexts, user modalities, and domains, including text and natural language [9], videos [10], audio [11], and games [4], [12]. Video games present their own rich and unique form of human-computer interaction [13] in that they allow the user to play an active role during consumption and encompass multiple modalities. Traditionally, models of affect learn to map one or more of these input modalities (e.g., facial input [14], pixels [15], game states [6]) to an affect label reported by a human annotator using supervised learning. Reliably collecting such labels for affect, however, is a significant challenge with a dedicated field of research within AC. Some modern data collection platforms, such as CARMA [16] and the PAGAN framework [17], enable the collection of such labels in real-time. Recent tools such as RankTrace [18] allow for the collection of unbounded time-continuous signals, which can be used, in turn, for regression or classification tasks, or converted into ordinal labels for preference learning methods [19].

Due to the several challenges involved in the collection of reliable affect labels, an alternative practice in AC is the analysis of an existing affective corpus. Standardized Emotion Elicitation Databases (SEEDs), allow the study of emotion by replicating real life in controlled settings. SEEDs have been solicited through multiple affect elicitors including images [20]–[22], videos [23]–[25], and even 3D objects [26]–[28]. In this study, we use the AGAIN dataset [6], comprising $1,100$ in-game videos and self-reported annotations of arousal across 124 participants and 9 games. The data consists of in-game telemetry, synchronised with video recordings of the participants' gameplay and time-continuous arousal signals using RankTrace [18]. Models trained on affect labels from the *Solid Rally* racing game of AGAIN have already shown promising results on predicting arousal by solely relying on pixel and in-game audio [15] but also training of human-like game playing agents [29].

In this paper, we build on earlier studies [29], [30] and extend the functionality of affect models for the purpose of generating environments that are tailored toward desired affect patterns. We thus attempt to close the affective game loop [4] through the game-level generation capacities of the algorithms introduced here.

B. PCG for Affective Computing

Since its inception a few decades ago, PCG has evolved to be a hugely influential and critical area of research within the domain of generative media. The initial focus of PCG methods was on adding replayability and novelty to games, such as by generating infinite levels in *Rogue* (Epyx, 1980), which spawned an entire genre of games referred to as *Rogue-likes* [31]. With the recent advent of large language models, the scope for generative systems in games has expanded substantially [32]. Large language models such as GPT have proven themselves capable of both generating text based on emotions [33] and processing and recognizing emotions [9]. Whilst affect-driven generation models for other domains remain in its infancy [34], such methods have shown promise in domains such as music generation [35], facial expression transformation [36], as well as dialogue generation [37].

Within games, affect-based generation has been instantiated by and mostly been explored through the *experience-driven PCG* (EDPCG) framework [2]. EDPCG aims to generate content that elicits a particular player experience when played. EDPCG primarily takes the form of a search-based PCG method [38], which generates content through algorithmic means such as local search or evolutionary search, allowing for more complex outputs given the right fitness function. Early examples of EDPCG applied in games involve generating personalised levels using some theory-based metric, such as generating racetracks according to *Koster's* [39] fun metric [40], and evolving interesting first-person shooter maps that maximize the duration of close fighting between players [41]. Shaker et al. also generated levels for *Super Mario Bros.* (Nintendo, 1985) according to predicted states of the player including engagement, frustration, and challenge [42]. More recent examples of EDPCG include the generation of video game levels for the *Sonancia* [43] system according to the tension experienced by players [44].

Recently, EDPCG was combined with PCGRL [45] to generate game levels for *Super Mario Bros* (Nintendo, 1985) [7] using a reward function which moderates diversity, inspired by *Koster's* principle of *fun* [39]. The EDRL framework, in short, has also been extended to generate levels using an episodic soft-actor critic algorithm, allowing it to better tailor itself to individual players [46]. To our knowledge, however, EDRL has yet to tackle affect-aware generation using human affect annotations in a continuous manner. This paper introduces the first EDRL agent that relies on time-continuous affect models that, in turn, allow an RL designer to incrementally build content according to a target player type and target affect pattern. Our introduced EDRL agent can either act as a novel-level generator or an affect-aware design assistant.

III. Experience-Driven Content Generation

In this paper, we propose a novel approach for generating video game content using a model of human affect demonstrations. The algorithm builds upon the EDRL [7] framework by using a data-driven approach for evaluating generated levels through an evaluator agent combined with an affect model, shown in Fig. 2. We describe our approach in detail in Section III and the different reward functions tested in Section III-C.

A. Designer

The designer is responsible for generating the stimuli (racetrack in case of *Solid Rally*) that will be passed to the evaluator agent (hereafter the *evaluator*) during optimisation. Whilst the designer can use any generation method (e.g., PCGML [47], PCGRL [45]), we test two implementations in this paper: one which takes the form of a search-based generator using a evolutionary RL method (i.e. genetic algorithm), and an RL designer using a variant of the Go-Explore algorithm [48] called Go-Blend [29] (see Section IV-B). Regardless of implementation, the output of the designer is a level (L) that is assigned a reward based on a function used by the evaluator. Based on this loop of generating new levels followed by feedback from the evaluator's simulation data, the designer should be able to optimise its output over time. Once complete, it should be capable of generating new stimuli that satisfy the target affect trace for a particular player type.

B. Evaluator

The role of the evaluator is two-fold. First, the evaluator ensures the feasibility of the generated stimuli (i.e., the generated racetrack) if required during training. If the evaluator identifies a stimulus as infeasible (e.g., an unplayable level featuring a track intersecting itself), the individual is penalised to discourage the designer from re-using this design. Second, the evaluator is responsible for generating the state (S_L) and affect (A_L) traces for the given stimulus. In this paper, we test an evaluator that generates traces by simulating the playback of the stimuli using an AI agent.

One more critical design decision for the evaluator is its behaviour during simulation. Due to the data-driven nature of this framework, the affect model and behaviour evaluator can be tailored to the human demonstrations of a specific player type through clustering. Training the affect model on a specific cluster (i.e., player type) will push the system to generate tracks that are tailored to their specific affective patterns. The specifics of the implementation of the evaluator are highly dependent on the environment and the type of content being generated, as discussed in Section IV-B.

C. Reward Function

The reward function is the final important component of our framework, which guides the designer to generate the desired stimuli. For example, if the generator is required to generate content that maximises arousal, it must be trained to produce stimuli that elicit A_L close to the maximum arousal value. Beyond affect, the reward function can also be used to reward the similarity of state traces (i.e., in the behaviour of the evaluator). Generally speaking, the reward function, denoted by R_L, measures the similarity, D between a generated affect trace (A_L) and the desired affect trace (A_T) as follows:

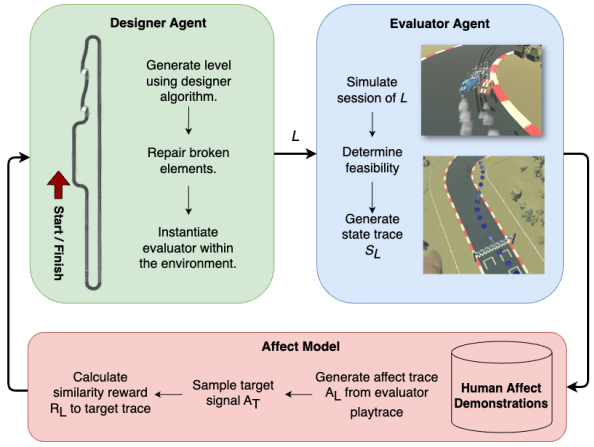

Fig. 2. High-level overview of our EDRL framework, with visual examples from our case study detailed in Section IV-B. L refers to the level generated by the designer, R_L refers to the reward function (Eq. 1), S_T and A_T are the state and affect traces generated by the evaluator during testing, and A_T is the target affect signal the agent is trying to imitate.

$$R_L = -D(A_L, A_T) \tag{1}$$

Similarity can be measured using any distance metric, D, considered appropriate for each use case. In this paper, we use the *area between curves* method outlined in [49] as it is robust to noise and outliers. To reward for similarity, we then assign a negative value based on the distance (i.e., the greater the distance from A_T, the harsher the penalty) as seen in Eq. 1. It is important to note that the reward function proposed here can be used by either a traditional RL method (as a reward) or an evolutionary RL method (as a fitness function).

IV. Arousal-Driven Racetrack Generation

In this section, we briefly describe our case study platform, the *Solid Rally* racing game from the AGAIN dataset [6], along with an overview of our implementation (Section IV-B) and our arousal model (Section IV-A). *Solid Rally* is a 3D real-time rally driving game built on the Unity game engine. The player must race against three other opponent cars around a racing circuit featuring corners, straights, loops, and bridges (see Fig. 3). As the cars drive around the racetrack, they are awarded points for passing through checkpoints located at the end of each component they drive through. The game features two sets of controls, one for steering (left, right, straight) and one for the gas pedal (forward, backward, neutral). The races have a maximum time limit of three laps, or 2 minutes if the player fails to finish before the time limit.

A. Arousal Model for Solid Rally

In this section, we describe the model of arousal used for evaluating the quality of the generated racetracks. As

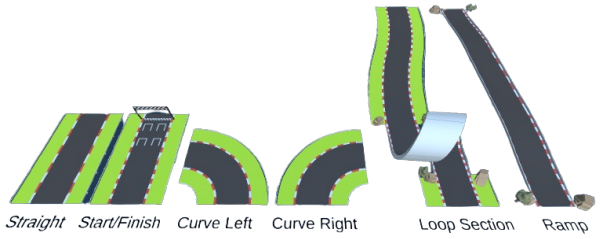

Fig. 3. Visualisation of the different possible track components found in *Solid Rally*.

mentioned above, this system is built on top of *Solid Rally*, a game from the AGAIN dataset [6] containing over 100 human demonstrations of continuous arousal traces.

The observation data of the model consists of a set of 29 game features returned from the game engine about the cars' speed, score, distance to opponents, and various other in-game metrics. Each record in the dataset corresponds to a 3-second time window where the above features and arousal values are averaged. We then convert the records into preferences by comparing pairs of consecutive windows and giving them an ordinal affect label as denoted by the difference between their arousal values: increase, decrease, or stable arousal [19]. Inspired by earlier studies [15], [29] we use a preference threshold of 0.15, which means that for a time window to be labelled as an increase or a decrease, the corresponding absolute change must be greater than 0.15. This threshold helps us combat reporting biases across subjects in the affect annotations provided. Finally, the dataset is normalised using min-max normalisation to ensure all the features lie within $[0, 1]$.

To better differentiate between different annotator types, we cluster the dataset based on the player's score and arousal traces. We perform clustering by computing the distance between all possible pairs of annotators using the area between curves method described in Section III-C. Doing so yields three clusters of annotators, two of which are a group of high-performing players with a substantial difference in their annotated arousal traces. We label these two clusters as *Excited* experts and *Unexcited* experts. Figure 4 shows the mean arousal traces of these clusters, where we can clearly observe that *Excited* experts have a constant increase in arousal over time, whereas *unexcited* experts have a sharp rise at the start, followed by a gradual descent as players of this cluster get less and less aroused as gameplay progresses. The third cluster we identified hosts a group of poor-performing players in terms of behaviour, which we call *Beginners*.

For the affect model, we implement a distance-weighted KNN model [8] to generate the change in arousal from a given pair of states, in this case, the evaluator's current state and the previous one. The KNN takes as input two state vectors (i.e., 29 game features across 3-second windows each as described above) and searches the corpus of affect annotations for the closest K entries using a pairwise Euclidean distance measure. We down-sample the dataset to only include the samples with

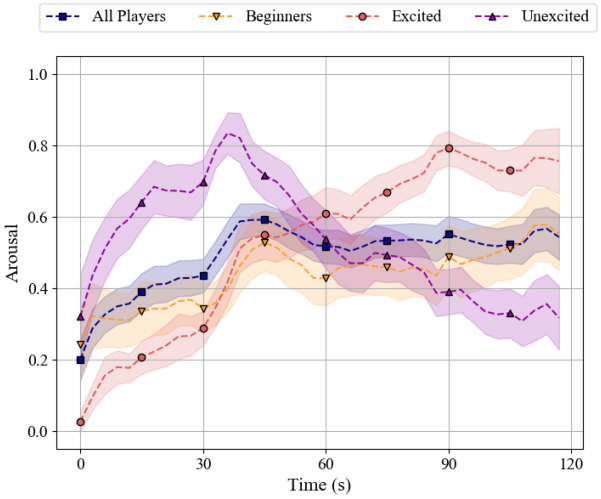

Fig. 4. Mean arousal traces from all players and the three identified clusters of players within *Solid Rally*, which we call the *Beginners*, the *Excited Experts* and *Unexcited Experts*. Shaded areas denote a 95% confidence interval.

changes in arousal (i.e., ignore stable entries) to focus on the most informative data samples during evaluation. Entries with an arousal increase are labelled with 1, whereas arousal decreases are labelled with 0. Once the K nearest neighbours are found, we use a weighted average (i.e., the closer to the current state, the higher the weight) to generate an arousal label for the current state. This label lies in $[0, 1]$ and can be interpreted as the change of arousal for the current state with respect to the previous state. The closer to 0 or 1, the more confident the agent is in its prediction of the change.

B. Arousal-Driven PCG for Solid Rally

In this paper, we implement a generator using two designer approaches, the first is an evolutionary algorithm, often seen as an EDPCG approach, and the second is an EDRL agent using a variant of the Go-Explore algorithm [48] called Go-Blend [29]. To accomplish this task, a framework had to be built around *Solid Rally* to facilitate the creation of new racetracks, and communication between the internal game state and the generator's code. The game was converted into an Open-AI Gym environment using the Unity-Gym package, which allows for the training of Unity ML agents [50] and custom agents using Python.

Racetracks are represented as a string of track components, the types of which are visualised in Fig. 3. Each component has a start position and end position, which are used to connect the pieces during generation. The components are scaled to be the same unit size, i.e., the straight, start/finish, and curves are all 1 *tile* long, whereas the loop and ramp are 3 *tiles* long. This tile-based formatting allows us to represent the environment as a 2-dimensional grid of tiles, simplifying the process of detecting collisions and determining feasibility. The grid is initialised to contain zeros (i.e., empty tiles) at the start of training.

1) EDPCG Designer: Our implementation of the EDPCG designer for this case study is as follows. The genotype is a

string of IDs of a fixed target length, which are mapped to the components and instantiated in sequence in the environment. The phenotype is generated by iteratively replacing the component ID found in the genome with the corresponding track piece and updating the 2D grid accordingly. Collisions are detected by checking whether the current location already has a component (i.e., an ID $\neq \emptyset$) before placing the next piece. If the racetrack is found to collide with itself, the stimulus is considered infeasible and is assigned a high penalty value (i.e., the worst possible reward of -1000).

We employ a genetic algorithm to allow the EDPCG designer to search the solution space for highly fit tracks. Our implementation follows the $(\mu - \lambda)$ evolutionary strategy [51], with the addition of a crossover operation during reproduction. This works by first initialising a population consisting of random individuals. Then, across many generations, we pass each individual to the evaluator if they are considered feasible, and assign them a fitness as the reward R_L (see Eq. (1)) based on their generated A_L. After the entire population is evaluated, we select the μ best individuals to act as parents for the next generation. The next generation of individuals is created by repeatedly selecting pairs of parents and using one-point crossover and mutation to generate offspring. We use uniform mutation, meaning that we randomly change components of an individual to a different ID with a small chance set by the mutation rate. We preserve the best individual between generations (elitism) to ensure the population retains feasible and high-quality individuals after reproduction. Once this process is completed (i.e., the maximum number of generations is reached), the designer outputs a set of high-quality tracks that should elicit the desired affect pattern.

The hyperparameters for the EDPCG designer in this use case are selected based on preliminary experiments and are as follows. We use a parent population size (μ) of the best 10 individuals from the current population. Our offspring population size (λ) was set to 50 individuals, with the constraint that each individual in the population must be unique (i.e., when a new individual is generated, it is only inserted into the population if an identical copy does not already exist). We use one-point cross-over and mutation, with a mutation rate of 10% during reproduction. Parents are randomly chosen for reproduction from the μ best individuals. We ran 10 separate runs of 50 generations per scenario described above, and present the average results across runs, including the 95% confidence interval.

2) EDRL Designer: As an alternative to the search-based generator, we use Go-Explore as the EDRL designer algorithm for this experiment, as it has proven itself a strong candidate for exploring deceptive and challenging environments. This designer follows the same collision rules as the EDPCG designer, but is driven by a different optimisation process. The EDRL designer iteratively builds racetracks using the Go-Explore exploration loop. First, a cell is sampled from the archive and its state is produced in the environment by repeating its trajectory of actions. From there it explores a single action, assigns a reward (R_L; see Eq. (1)), and checks whether to store it in the archive or not. Cells are stored if they are unique, or if they have a larger reward than the existing cell. Our cells' state representation is a state vector containing the number of straights, turns, loops, bridges, and the length of the Dijkstra path [52]; described in the following section. This more abstract representation was chosen to group similar racetracks into the same cell and only separate meaningfully different layouts, resulting in a smaller archive and better search space exploration. Finally, we cap the cells' trajectory length to a maximum of 10 pieces to maintain a fair comparison with the EDPCG and Random designers, meaning that Go-Explore cannot explore a cell if it already contains 10 actions. Once complete, we take the best cell from the archive with 10 actions as our elite individual.

3) Playable Racetracks via Dijkstra: Since the tiles placed by the designer are not guaranteed to create a racetrack that forms a circuit (i.e., a connected start and finish tile), the PCG algorithms are required to find the shortest path between the first and last tile, connect them, and form a playable track. With this in mind, we use the well-established Dijkstra algorithm [52] to search the grid for the shortest path between the start and the end tile. Once the shortest path is found, Dijkstra's algorithm places appropriate tiles to fill the path and complete the circuit. Dijkstra is only allowed to use simple tiles (i.e., straight, curve left, curve right) to implicitly encourage the designer to not rely heavily on this method and, preferably, close the circuits through its own designs. In case no path can be found using Dijkstra—due to the absence of a feasible route between the start and the end tile—the track is marked as infeasible and is discarded.

4) Evaluator: The evaluator consists of an in-game agent that drives around the generated racetracks and queries the arousal affect model described in Section IV-A. The agent's behaviour is governed by a checkpoint system that forms part of the original *Solid Rally* implementation, which works as follows. Each component of the generated level has a checkpoint placed at its endpoint. The agent drives such that it minimises the distance to the next checkpoint. When the agent drives through this checkpoint, it is given an increment to its score, and its target is set to the next checkpoint. This simple system allows the agent to behave reliably across any kind of level generated. This is the same system used by the opponent cars which the evaluator is racing against and—like the original game—the race takes place over 3 laps. As the evaluator simulates a race, it queries the KNN affect model every 3 seconds of in-game time. Note that due to the relatively deterministic nature of the evaluations, we only perform one simulation per track; however, in less deterministic game environments averaging the R_L across multiple independent runs would be necessary for yielding more stable training signals.

V. Experimental Protocol

We evaluate generators by testing them across three different scenarios by which they must generate content that elicits a desired (i.e., target) affect trace to a game-playing agent. As

mentioned in Section III-C, we use the area between curves as the distance measure between the generated affect trace (A_L) and the target trace (A_T). The content was tested by simulating three laps around the racetrack to remain consistent with the original dataset. The first scenario is the *Minimise Arousal* experiment, where the generator must create a racetrack that causes the agent's arousal to constantly decrease over time. The *Maximise Arousal* scenario mirrors the former by requiring the agent's arousal to constantly increase over time. Finally, the *Fluctuating Arousal* requires the racetrack to elicit an arousal trace that varies over time. In particular, the level must first maximise arousal for its first third, then minimise arousal for the second third, and finally, for the last third of the track, maximise arousal for the player. We compare our affect-driven level designers against a baseline random designer agent, which places a series of random tiles and then uses Dijkstra to close the circuit.

To evaluate the generators, we pick the best individual based on the *accuracy* of the output arousal signal elicited from the generated track to the target signal. Specifically, we treat the arousal modelling of the generated track as a binary classification problem, and we measure the rate at which the output signal and the target signal agree on the change in affect (i.e., increase versus decrease). Since our arousal model outputs a value between 0 and 1, an increase (or decrease) is indicated by any output over (below) 0.5. An accuracy of 100% (or 0%) means that the generated arousal trace and the desired arousal trace agree (disagree) completely on the change of arousal across the complete racetrack. Comparisons are made across 10 runs for each experiment configuration. We report average accuracy values and corresponding 95% confidence intervals. We denote significance when we observe non-overlapping confidence intervals between experiments at the $p < 0.05$ level. Beyond this, we also perform an expressive range analysis on the output of the designers [53]. The expressive range analysis offers us complementary qualitative insights about the variation of the components used on the racetrack for each target signal.

VI. Results

The results of our experiments comparing the three different track designers can be seen in Table I: we test each designer across four different player clusters and three target arousal scenarios. As expected, we observe that both the EDPCG and EDRL designers outperform the random designer (significantly based on the 95% confidence intervals) across all player clusters and scenarios. It is also clear from the results that certain scenarios were more challenging than others across player clusters. For example, the *excited* and *unexcited experts* produce the most varied results across the three different scenarios tested. *Unexcited experts* are easier to satisfy as a player group when we attempt to maximise their arousal with accuracy values over 95% for both EDPCG and EDRL. The opposite holds for the *excited experts* and the *beginners*, as it seems that minimising arousal is easier than maximising arousal for these player groups. Experiments with *all players* available in our corpus yields overall more consistent results across designers and scenarios (i.e., accuracy values in between 68% and 87%) due to the larger and more diverse set of human demonstrations. This larger and more diverse corpus most likely allows the designer to explore a larger variety of arousal states and therefore yield more consistent performances.

TABLE I

Accuracy of the best individual from each of the three scenarios (minimise, maximise or fluctuate arousal), grouped by designer algorithm (random, EDPCG, EDRL) and averaged across 10 runs, including 95% confidence intervals. Values in bold indicate the highest accuracy value obtained across designers for a particular scenario. Underlined and bold values indicate statistically significant differences across designers.

	All Players	**Beginners**	**Excited**	**Unexcited**
Random				
Max. Arousal	72.4 ± 2.8	58.0 ± 2.6	50.7 ± 3.5	89.8 ± 1.0
Min. Arousal	66.4 ± 2.5	76.9 ± 1.7	89.2 ± 1.9	35.8 ± 3.6
Fluctuating	57.1 ± 1.7	72.3 ± 3.6	56.9 ± 1.6	63.5 ± 2.0
EDPCG				
Max. Arousal	84.4 ± 2.0	$\mathbf{78.8 \pm 2.0}$	70.4 ± 7.7	95.9 ± 0.9
Min. Arousal	$\mathbf{81.6 \pm 2.2}$	85.8 ± 2.1	96.6 ± 1.0	42.8 ± 2.6
Fluctuating	68.3 ± 2.2	86.3 ± 1.0	67.2 ± 3.0	71.1 ± 1.6
EDRL				
Max. Arousal	$\mathbf{87.7 \pm 2.8}$	$\mathbf{78.8 \pm 3.4}$	$\mathbf{75.9 \pm 3.9}$	$\mathbf{97.4 \pm 0.8}$
Min. Arousal	$\mathbf{81.6 \pm 2.9}$	$\underline{\mathbf{96.2 \pm 1.5}}$	$\mathbf{98.1 \pm 1.2}$	$\underline{\mathbf{53.4 \pm 2.8}}$
Fluctuating	$\underline{\mathbf{74.7 \pm 2.5}}$	$\mathbf{87.5 \pm 2.4}$	$\underline{\mathbf{75.3 \pm 2.1}}$	$\mathbf{72.7 \pm 0.9}$

We argue that the varying performance of the tested methods across player types is due to their dissimilar affective patterns (see Fig. 4). We argue that the inherent subjectivity of reported affect as elicited by the layout of the circuit causes such dissimilarities. For example, compared to other player types, the arousal changes of the *beginner* players appear to be more consistent to the design of the circuit (i.e., the context). Consistent affect demonstrations, in turn, make the task of content generation far easier as predicting any target arousal trace is much simpler. On the other end of the spectrum, the *unexcited experts* generally showcase less arousal variation with respect to racetrack changes. We argue that this is due to habituation effects, as this group of expert players appears to be stimulated less after driving their first lap. Another potential reason for the observed differences is that the skill level of the player clusters had a strong effect on the annotated arousal levels. Specifically, our evaluator agent showcases only marginally better performance to the opponent AI cars, which is roughly equivalent to the behaviour of the *beginners* cluster. Such an agent cannot replicate how either of the expert groups would play in any generated racetrack, and it thus results in skewed arousal levels and larger performance discrepancies across scenarios for these groups.

As seen in Table I, EDRL yields superior performances compared to the random baseline while it matches or outperforms the performance of EDPCG across nearly all scenarios and player types. Results show that EDRL is, on average, able

to significantly outperform EDPCG in 4 out of the 12 scenarios explored and yields higher performances in 10 out of those scenarios. Intuitively, we argue that the EDRL designer is able to better explore the search space over time, whilst the EDPCG designer is more prone to converging to local optima through the simple evolutionary strategy implemented.

Finally, we also analyse the differences in track layouts across the scenarios we tested. Unsurprisingly, tracks built to minimise arousal appear to contain significantly more basic tiles (i.e., straights and turns) compared to tracks built to maximise arousal, with averages of 24.3 ± 1.6 and 19.7 ± 2.5 respectively. We observe the inverse phenomenon when we attempt to maximise arousal as generators use more complex event tiles (i.e., bridges and loops) to induce increasing arousal whenever possible; an example of this phenomenon can be seen in Fig. 1. Placing more event tiles in the maximise arousal scenario—on average 5.1 ± 0.9 tiles compared to 3.8 ± 0.5 tiles in the minimise arousal scenario—increases the probability of driving errors (e.g., driving off-road, crashing into other cars, barriers) from our evaluator agent. Such driving style can, in turn, contribute to increases in arousal elicitation.

VII. Discussion

The proposed approach is novel in that it generates affect-aware content in a continuous manner via the EDRL framework. Importantly, the generated content can be tailored to a target affective pattern or trace for a particular user. Our results show that EDRL is capable of matching and sometimes outperforming EDPCG when it comes to designing racetracks with a specific arousal trace as a target. Incorporating Go-Explore's [48] robustification phase would allow our EDRL agent to function in an online capacity. This superior performance paired with the ability of EDRL to act as an online content generator (e.g., as in [2], [46]) highlight the promise of this approach in a real-world setting. Future investigations employing more complex reward functions, tested across dissimilar game genres and other domains, will help us validate the potential of EDRL for affect-driven generation in a general fashion. While tested initially in the domain of games, the EDRL method introduced here is applicable to any affective interaction task that envisions closing the affective loop via content generation mechanisms. As EDRL has shown promise to adapt itself to clusters of users and target affect patterns in this paper, we envision methods relying on the EDRL principle to be able to cope with dynamic shifts of user preferences and behaviours over time.

One of the limitations of the current approach lies in the simple behaviour of the evaluator agent. Whilst we chose the checkpoint system that governs its behaviour—due to its reliability to generalise to new levels—the resulting behaviour of the agent is very deterministic and does not follow human-like patterns of play. More specifically, the agent does not factor in opponent cars, so it is prone to collide with them in situations that most human players would likely avoid. The agent is also constantly accelerating (a behaviour that most human players would likely not follow) especially if they get stuck or need to slow down for a sharp turn. Consequently, since the agent tends to play through components with very similar behaviour, it limits the generator's ability to thoroughly explore the solution space as the full range of possible scenarios per component is not sufficiently explored. One way of addressing this limitation in the future is to train agents that behave similarly to the target clusters, thus having a more representative evaluator in terms of behaviour and, therefore, more reliable affect-driven generation.

Another limitation of this work lies in the generalisability of our arousal model. Since the dataset used in this paper only contains annotations and telemetry from sessions on a single-track layout, the arousal model does not have a large distribution of contextual data available, which in turn limits its performance when assessing new content. As a result, whilst we show that EDRL is capable of generating content using this arousal model, it remains to be seen whether these results will accurately reflect human feedback. This is an important avenue for future work, as quantitatively correlating the generated tracks via a human study (e.g., in [54]) will confirm that the generator is capable of matching content to human emotions.

Finally, collecting more human (behavioural and affect) demonstrations on a diverse set of new racetracks will help us build more general representations between arousal, playing behaviour and game content. Yet another approach could be a mixed initiative one [55], where a human user provides feedback on the generated tracks to help steer affect-driven optimisation in the right direction. This approach could gradually help us build a more personalised model of affect, and—with enough data obtained—could remove the need for running simulations and instead learn to generate affect directly from the stimuli presented.

VIII. Conclusions

We introduced a novel framework that expands current experience-driven content generation frameworks [2], [7] able to generate content that elicits tailor-made continuous affective patterns to a particular type of user. We test the capacity of two such generative methods—one based on a genetic algorithm and a second one driven by RL exploration—to create affect-aware and personalised racetrack levels in games. Our results demonstrate that both approaches can accurately match a number of desired affect traces for certain types of players, but the EDRL method appears to be more efficient and robust. Both methods, however, face challenges to elicit certain affect patterns (e.g., maximise arousal throughout the level) to player types that have not experienced such patterns through their demonstrations (e.g., low-aroused beginner players). Findings of this paper suggest that it is possible to continuously generate affect-driven content and, importantly, it appears that the proposed EDRL method can create personalised and tailor-made content for various user types. While the results of this paper are specific to game content generation, the methods proposed are directly applicable to any affective interaction domain in need of personalised content creation.

Ethical Impact Statement

This paper makes use of an existing dataset (AGAIN dataset [6]) of human demonstrations collected from crowd workers on the Amazon Mechanical Turk (mTurk) platform. This dataset is publicly available, and participants gave their consent for their data to be stored and utilised in an anonymous fashion. To the best of our knowledge, there is no significantly negative application of the methods we use in this paper and no added privacy or discrimination risk. The data used in the paper and environment are publicly available for scientific reproducibility and for further extensions of this study.

References

[1] R. W. Picard, *Affective computing*. MIT press, 2000.

[2] G. N. Yannakakis and J. Togelius, "Experience-driven procedural content generation," *Transactions on Affective Computing*, vol. 2, no. 3, pp. 147–161, 2011.

[3] K. Höök, "Affective loop experiences–what are they?," in *Persuasive Technology Conference*, Springer, 2008.

[4] G. N. Yannakakis and D. Melhart, "Affective game computing: A survey," *Proceedings of the IEEE*, 2023.

[5] E. Hudlicka, "Affective computing for game design," in *Conference on Intelligent Games and Simulation*, pp. 5–12, McGill University Montreal, 2008.

[6] D. Melhart, A. Liapis, and G. N. Yannakakis, "The arousal video game annotation (again) dataset," *Transactions on Affective Computing*, vol. 13, no. 4, pp. 2171–2184, 2022.

[7] T. Shu, J. Liu, and G. N. Yannakakis, "Experience-driven pcg via reinforcement learning: A super mario bros study," in *Conference on Games*, IEEE, 2021.

[8] L. E. Peterson, "K-nearest neighbor," *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.

[9] J. Broekens, B. Hilpert, S. Verberne, K. Baraka, P. Gebhard, and A. Plaat, "Fine-grained affective processing capabilities emerging from large language models," in *Conference on Affective Computing and Intelligent Interaction*, IEEE, 2023.

[10] J. M. Girard, Y. Tie, and E. Liebenthal, "Dynamos: The dynamic affective movie clip database for subjectivity analysis," in *Conference on Affective Computing and Intelligent Interaction*, IEEE, 2023.

[11] J. Kim and E. André, "Emotion recognition based on physiological changes in music listening," *Transactions on pattern analysis and machine intelligence*, vol. 30, no. 12, pp. 2067–2083, 2008.

[12] G. N. Yannakakis and A. Paiva, "Emotion in games," *Handbook on affective computing*, vol. 2014, pp. 459–471, 2014.

[13] P. Barr, J. Noble, and R. Biddle, "Video game values: Human–computer interaction and games," *Interacting with Computers*, vol. 19, no. 2, pp. 180–195, 2007.

[14] Y. Guo, Y. Xia, J. Wang, H. Yu, and R.-C. Chen, "Real-time facial affective computing on mobile devices," *Sensors*, vol. 20, no. 3, p. 870, 2020.

[15] K. Makantasis, A. Liapis, and G. N. Yannakakis, "The pixels and sounds of emotion: General-purpose representations of arousal in games," *Transactions on Affective Computing*, vol. 14, no. 1, pp. 680–693, 2021.

[16] J. M. Girard, "Carma: Software for continuous affect rating and media annotation," *Journal of open research software*, vol. 2, no. 1, 2014.

[17] D. Melhart, A. Liapis, and G. N. Yannakakis, "Pagan: Video affect annotation made easy," in *Conference on Affective Computing and Intelligent Interaction*, pp. 130–136, IEEE, 2019.

[18] P. Lopes, G. N. Yannakakis, and A. Liapis, "Ranktrace: Relative and unbounded affect annotation," in *Conference on Affective Computing and Intelligent Interaction*, pp. 158–163, IEEE, 2017.

[19] G. N. Yannakakis, R. Cowie, and C. Busso, "The ordinal nature of emotions: An emerging approach," *Transactions on Affective Computing*, vol. 12, no. 1, pp. 16–35, 2018.

[20] P. J. Lang, M. M. Bradley, B. N. Cuthbert, *et al.*, "International affective picture system (iaps): Instruction manual and affective ratings," *The center for research in psychophysiology, University of Florida*, 1999.

[21] A. Marchewka, Ł. Żurawski, K. Jednoróg, and A. Grabowska, "The nencki affective picture system (naps): Introduction to a novel, standardized, wide-range, high-quality, realistic picture database," *Behavior research methods*, vol. 46, pp. 596–610, 2014.

[22] B. Kurdi, S. Lozano, and M. R. Banaji, "Introducing the open affective standardized image set (oasis)," *Behavior research methods*, vol. 49, pp. 457–470, 2017.

[23] S. Carvalho, J. Leite, S. Galdo-Álvarez, and O. F. Gonçalves, "The emotional movie database (emdb): A self-report and psychophysiological study," *Applied psychophysiology and biofeedback*, vol. 37, pp. 279–294, 2012.

[24] Y. Baveye, E. Dellandrea, C. Chamaret, and L. Chen, "Liris-accede: A video database for affective content analysis," *Transactions on Affective Computing*, vol. 6, no. 1, pp. 43–55, 2015.

[25] M. Gnacek, L. Quintero, I. Mavridou, E. Balaguer-Ballester, T. Kostoulas, C. Nduka, and E. Seiss, "Avdos-vr: Affective video database with physiological signals and continuous ratings collected remotely in vr," *Scientific Data*, vol. 11, no. 1, p. 132, 2024.

[26] D. Popic, S. G. Pacozzi, and C. S. Martarelli, "Database of virtual objects to be used in psychological research," *Plos one*, vol. 15, no. 9, p. e0238041, 2020.

[27] D. Peeters, "A standardized set of 3-d objects for virtual reality research and applications," *Behavior research methods*, vol. 50, pp. 1047–1054, 2018.

[28] J. Tromp, F. Klotzsche, S. Krohn, M. Akbal, L. Pohl, E. M. Quinque, J. Belger, A. Villringer, and M. Gaebler, "Openvirtualobjects: An open set of standardized and validated 3d household objects for virtual reality-based research, assessment, and therapy," *Frontiers in Virtual Reality*, vol. 1, p. 611091, 2020.

[29] M. Barthet, A. Khalifa, A. Liapis, and G. Yannakakis, "Generative personas that behave and experience like humans," in *Foundations of Digital Games Conference*, ACM, 2022.

[30] M. Barthet, A. Khalifa, A. Liapis, and G. N. Yannakakis, "Play with emotion: Affect-driven reinforcement learning," in *2022 10th International Conference on Affective Computing and Intelligent Interaction (ACII)*, pp. 1–8, IEEE, 2022.

[31] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup, "Procedural content generation for games: A survey," *Transactions on Multimedia Computing, Communications, and Applications*, vol. 9, no. 1, 2013.

[32] R. Gallotta, G. Todd, M. Zammit, S. Earle, A. Liapis, J. Togelius, and G. N. Yannakakis, "Large language models and games: A survey and roadmap," *arXiv preprint arXiv:2402.18659*, 2024.

[33] K. Schaaff, C. Reinig, and T. Schlippe, "Exploring chatgpt's empathic abilities," in *Conference on Affective Computing and Intelligent Interaction*, IEEE, 2023.

[34] G. Nie and Y. Zhan, "A review of affective generation models," *arXiv preprint arXiv:2202.10763*, 2022.

[35] K. Miyamoto, H. Tanaka, and S. Nakamura, "Music generation and emotion estimation from eeg signals for inducing affective states," in *Conference on Multimodal Interaction*, pp. 487–491, 2020.

[36] R. Wu, G. Zhang, S. Lu, and T. Chen, "Cascade ef-gan: Progressive facial expression editing with local focuses," in *Conference on Computer Vision and Pattern Recognition*, pp. 5021–5030, IEEE, 2020.

[37] P. Colombo, W. Witon, A. Modi, J. Kennedy, and M. Kapadia, "Affect-driven dialog generation," *arXiv preprint arXiv:1904.02793*, 2019.

[38] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based procedural content generation: A taxonomy and survey," *Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 172–186, 2011.

[39] R. Koster, *Theory of fun for game design*. O'Reilly Media, Inc., 2013.

[40] J. Togelius, R. De Nardi, and S. M. Lucas, "Towards automatic personalised content creation for racing games," in *Symposium on Computational Intelligence and Games*, pp. 252–259, IEEE, 2007.

[41] L. Cardamone, G. N. Yannakakis, J. Togelius, and P. L. Lanzi, "Evolving interesting maps for a first person shooter," in *Applications of Evolutionary Computation - EvoGAMES*, pp. 63–72, Springer, 2011.

[42] N. Shaker, G. N. Yannakakis, J. Togelius, M. Nicolau, and M. O'neill, "Evolving personalized content for super mario bros using grammatical evolution," in *Artificial Intelligence and Interactive Digital Entertainment Conference*, vol. 8, pp. 75–80, AAAI, 2012.

[43] P. Lopes, A. Liapis, and G. N. Yannakakis, "Sonancia: Sonification of procedurally generated game levels," in *International Computational Creativity Conference*, 2015.

[44] P. Lopes, A. Liapis, and G. N. Yannakakis, "Framing tension for game generation," in *International Conference on Computational Creativity*, 2016.
[45] A. Khalifa, P. Bontrager, S. Earle, and J. Togelius, "Pcgrl: Procedural content generation via reinforcement learning," in *Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 16, pp. 95–101, ACM, 2020.
[46] Z. Wang, J. Liu, and G. N. Yannakakis, "The fun facets of mario: Multifaceted experience-driven pcg via reinforcement learning," in *Foundations of Digital Games Conference*, ACM, 2022.
[47] J. Liu, S. Snodgrass, A. Khalifa, S. Risi, G. N. Yannakakis, and J. Togelius, "Deep learning for procedural content generation," *Neural Computing and Applications*, vol. 33, no. 1, pp. 19–37, 2021.
[48] A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune, "First return, then explore," *Nature*, vol. 590, no. 7847, pp. 580–586, 2021.
[49] C. F. Jekel, G. Venter, M. P. Venter, N. Stander, and R. T. Haftka, "Similarity measures for identifying material parameters from hysteresis loops using inverse analysis," *International Journal of Material Forming*, vol. 12, pp. 355–378, 2019.
[50] A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar, *et al.*, "Unity: A general platform for intelligent agents," *arXiv preprint arXiv:1809.02627*, 2018.
[51] T. Bäck and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary Computation*, vol. 1, no. 1, 1993.
[52] E. W. Dijkstra, *A Note on Two Problems in Connexion with Graphs*, p. 287–290. New York, NY, USA: Association for Computing Machinery, 1 ed., 2022.
[53] G. Smith and J. Whitehead, "Analyzing the expressive range of a level generator," in *Workshop on procedural content generation in games*, ACM, 2010.
[54] G. N. Yannakakis and J. Hallam, "Real-time game adaptation for optimizing player satisfaction," *Transactions on Computational Intelligence and AI in Games*, vol. 1, no. 2, pp. 121–133, 2009.
[55] G. N. Yannakakis, A. Liapis, and C. Alexopoulos, "Mixed-initiative co-creativity," in *Foundations of Digital Games Conference*, ACM, 2014.