A Framework to Even-Out Racetrack Bias

Mirkamil Majid
The Ohio State University
Columbus, Ohio, USA
mierkamili.1@osu.edu

Roger Crawfis
The Ohio State University
Columbus, Ohio, USA
crawfis.3@osu.edu

Figure 1: Two Different Performing Cars Racing On A Balanced Racetrack.

ABSTRACT

Procedural Content Generation (PCG) has demonstrated its capability to create compelling game content across various domains, including racing games. In this paper, a novel approach utilizing PCG is presented that aims to generate racetracks with the primary objective of ensuring fairness and balanced gameplay regardless of the player's vehicle choice[1]. The proposed framework comprises three distinct phases: During the first phase, modular racetrack segments are procedurally generated in order to enhance track variety, which plays a crucial role in providing an engaging gaming experience. In the second phase, AI car controllers are employed on different vehicles to simulate driving through all generated racetrack segments, gathering various statistics. This step allows for the collection of data that will inform decision-making during the final assembly of the complete racetrack. Finally, in the third phase, the collected data is utilized to select and assemble the most suitable racetrack segments, ensuring fairness for all simulated vehicle types by avoiding any potential unfair advantages or disadvantages. To further support this innovative framework, its theoretical foundation is explored, and detailed explanations of each step involved in the process are provided.

[1]The pool of vehicles needs to be reasonable. A bicycle can never realistically compete against a NASCAR vehicle.

FDG 2024, May 21–24, 2024, Worcester, MA, USA

ACM ISBN 979-8-4007-0955-5/24/05
https://doi.org/10.1145/3649921.3659849

KEYWORDS

Racetrack Generation, Procedural Content Generation, Gameplay Balance, Racing, Tiling, Bias

ACM Reference Format:
Mirkamil Majid and Roger Crawfis. 2024. A Framework to Even-Out Racetrack Bias. In *Proceedings of the 19th International Conference on the Foundations of Digital Games (FDG 2024), May 21–24, 2024, Worcester, MA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3649921.3659849

1 INTRODUCTION

Procedural Content Generation (PCG) refers to the process of creating content through algorithms rather than manually designing them [19]. This approach allows for unlimited variations in content design, ensuring each generated in-game elements remains unique and captivating. By leveraging mathematical principles, a game can generate an infinite number of content layouts that provide varying levels of difficulty and excitement for players. In this paper, we explore the utilization of PCG for racetrack generation with a focus on maintaining balance within the gameplay experience.

In a racing game where all vehicles are identical and have the same specifications, there would be little to no need for balance or strategy in order to succeed. Instead, the race would purely come down to each racer's individual skill level - their reflexes, understanding of track layouts, and ability to handle pressure during high-stakes moments. However, when different vehicles with varying attributes are introduced into a racing game, it adds an entirely new layer to the competitive experience. These differences can range from speed boosters, handling, weight distribution or even weaponry on certain vehicles - all of which play a significant role in determining how a race will unfold and who ultimately emerges as the victor. This element of variety not only enhances strategic thinking but also adds an extra dimension to the gameplay experience that keeps players engaged for longer periods. However, if care is not taken in the construction of the racetrack and associated vehicle attributes, it can be difficult to avoid one vehicle class that dominates across player.

Our research work primarily focuses on game balance, which involves ensuring that all tracks are equally challenging and enjoyable for players regardless of their vehicle choice. By procedurally generating racetracks while maintaining fairness for all vehicles, we aim to enhance the overall gaming experience.

1.1 Problem Statement

Game balance is an integral component of game design, focusing on the degree of fairness and challenge provided by a game to its players [3]. Achieving optimal game balance requires an in-depth examination of various aspects, including game mechanics, player abilities, and preferences. In the context of racing games, essential factors for consideration include rational design principles such as race line, clipping points, track width, camber and height variation [12]. Additional elements to be considered may include multiple paths, track obstacles, and environmental effects. We define a fair racetrack as one that offers equal opportunities and challenges to all participants irrespective of their selected vehicle. ***Specifically, a racetrack is considered fair when each vehicle, operated optimally, completes the track in a similar amount of time.***

2 RELATED WORK AND CONTRIBUTION

Numerous studies have explored the development of algorithms for procedural racetrack generation. Similarly, research on gameplay balance has garnered significant interest. However, to the best of our knowledge, there has been limited investigation into the specific task of balancing racetracks based on race car attributes.

2.1 Procedural Racetrack Generation

Using search-based approaches, research has concentrated on creating a customized content, as suggested by Togelius et al. [22], resulting in tracks tailored to the player based on their profile. Other investigations have focused on enhancing the player fun factors by increasing the track diversity, which is determined by considering its curvature and speed profiles, as proposed by Cardamone et al. [6] and Maulidevi et al. [1]. Cardamone et al. later developed an interactive software that generates tracks using evolutionary algorithms [5]. Behrens et al. propose an algorithm for generating racetracks through the use of a control point representation, whereby these control points are procedurally positioned to create diverse tracks. [4]. Li et al. used pre-defined track blocks, which are pre-determined parameterized track segments, to generate roads in their research [10]. A more recent study put forth by Nascimento et al. uses chain code and images of pre-defined real tracks that achieves good results [13]. A study proposed by Gisslén et al. uses adversarial reinforcement learning techniques to generate racetrack with one agent being a builder and the other agent being a tester [8]. These studies often employ diverse methods to generate racetrack components and assemble them utilizing distinct algorithmic approaches.

2.2 Gameplay Balance

Analyzing gameplay balance can be highly subjective, influenced by numerous factors. A study conducted by Becker et al. [3] reveals the inconsistencies among authors' interpretations regarding this topic, emphasizing its complex nature. Despite these challenges, researchers have proposed various methods to approach game balance from their unique perspectives. For instance, Vicencio-Moreira et al. [23] introduced a scheme that adjusts aim assist for First-Person Shooter games. Tijs et al. [21] proposed to take users' overt behavior and physiological responses when considering balance adjustment. Jaffe [9], on the other hand, proposed simulating gaming experience using restricted artificial agents as an alternative approach. One study proposed by Cechanowicz et al. [7] explored ways to balance a racing game targeting different racers. These methods typically involve gathering data either through play-testing or simulation and apply their respective definitions of game balance accordingly.

2.3 Contribution

Our research contribution encompasses a three-step framework that combines constructive and constraint-based elements. Initially, we procedurally create racetrack segments. Subsequently, we employ AI agents to simulate gaming performance within these racetrack segments. Finally, leveraging the collected data, we generate a fair racetrack. To demonstrate the versatility of our framework, we explore alternative approaches for each step, elucidating the necessary conditions that such alternatives must meet for successful deployment.

3 FRAMEWORK OVERVIEW

Some racetracks, see Figure 2, can favor vehicles with certain attributes and provide an unfair advantage over others, creating an unbalanced game experience in a racing game with cars having widely different abilities. If the track has a lot of sharp turns, vehicles with high handling and acceleration attributes could have an edge over those with high-speed attributes. Similarly, if the track has a lot of straight stretches, vehicles with high-speed attributes may have an advantage over those with a low max speed.

Homogenizing vehicles with varying stats by making them perform identically throughout a race may result in uninteresting and monotonous experiences as it strips away each vehicle's distinct attributes. We propose a framework that involves creating racetrack loops with multiple segments where certain vehicles excel at some point, but their advantages are neutralized by the end. Each segment introduces drama as one vehicle may lead for a brief period before losing its lead on a different segment.

Formally, our framework is as follows

(1) Prepare swappable racetrack segments along with vehicles with multiple attributes.
(2) Simulate vehicles on racetrack segments to acquire performance data.

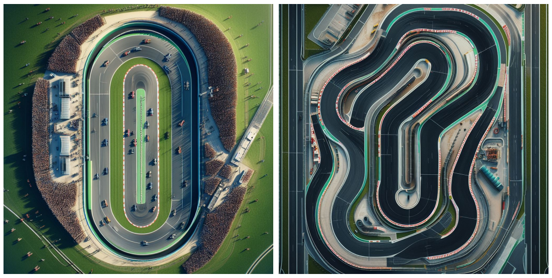

Figure 2: The racetrack on the left favors vehicles with high speed and the racetrack on the right favors those with better handling.

(3) Use acquired date to assemble a racetrack.

Numerous approaches can be employed to address each of the aforementioned steps. This research paper explores several potential methodologies as a proof of concept within the context of this framework.

4 VEHICLE ATTRIBUTES

Our research requires vehicles with varying abilities. Popular racing games, such as Mario Kart [24], present the following stats for their vehicles:

- Speed: Determines the top speed of the vehicle.
- Acceleration: Determines the rate the player can reach top speed
- Weight: Determines the impact when collision happens.
- Handling: Determines the ability to performs turns swiftly and accurately.
- Traction: Determines the interaction between the kart and environment.

One can always add or subtract from the list provided above to create different sets of stats. Our study employs a physics-based car controller with the following attributes: maximum speed, acceleration, maximum steering angle, and steering speed. While our current implementation adopts a simplified approach, future research endeavors can tailor these vehicle attributes to align with specific gameplay requirements.

5 PROCEDURAL RACETRACK SEGMENT GENERATION

We break down a racetrack circuit (loop) into smaller chunks which we call segments (similar to the work of Li et al.[10]). We want to be able to swap these segments with other segments to even our racetrack biases. Therefore, the alternative segments must have the same entry and exist point. This ensures seamless connectivity between segments during the swapping process. Ideally we would have a large set of segments and cars such that for each car there exists a segment where it is faster than all other cars.

Numerous methods exist for generating modular racetrack segments. These may include crafting them by hand or employing special techniques, such as Non-Uniform Rational B-Splines (NURBS) [15]. We deployed tile-based procedural content generation methods [11].

We first start with the common six-path tiles used in a tiling algorithm on a grid, see Figure 3. We then use these tiles as the racetrack segments and layout a racetrack tiling, as shown in Figure 4.

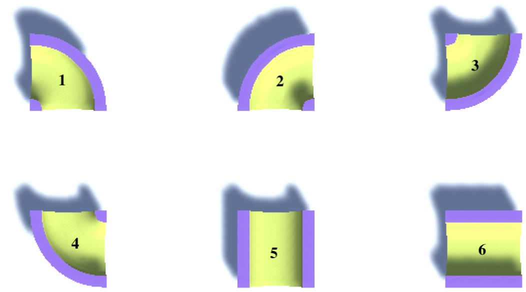

Figure 3: Six-path tiles: two straightaways, oriented either horizontally or vertically, and four turns that alter the track's direction. Tiles are numbered in black.

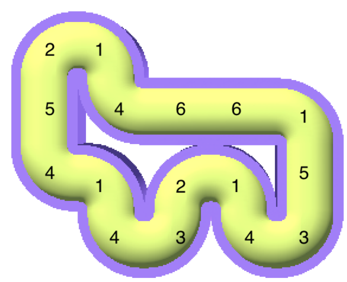

Figure 4: A racetrack generated using the six-path tiles. Tiles are numbered in black.

The issue with this approach lies in its restriction regarding the number of racetrack segments available, which proves insufficient for our desired application. To overcome this obstacle, we utilize the six-path tiles to construct larger tiles using tilings of these tiles.

In general, we need to construct multiple $n \times m$ tiling configurations that incorporate "straightaways" resembling tiles 5,6 from Figure 3 and "turns" resembling tiles 1,2,3,4. We have chosen to analyze all possible paths on five-by-five tilings where the exits are positioned at the center of each row or column. This generation process yielded a total of $24,918$ unique tiles. Some tiles are shown in Figure 5. These tiles provide a good diversity of racetrack segments to perform vehicle performance data collection and final racetrack assembly.

6 PERFORMANCE DATA COLLECTION

Using these generated racetrack segments, we can collect performance data for each vehicle. In an intensely competitive sports environment, it is crucial to identify and assess key performance

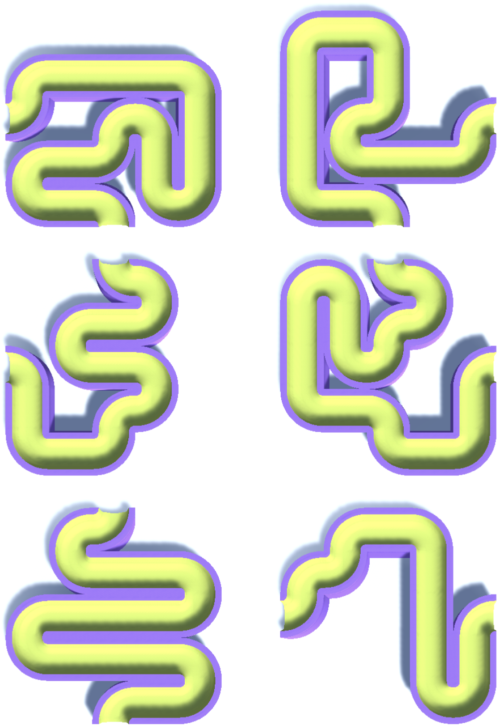

Figure 5: Six out of a possible 24, 918 **large tiles acquired form enumerating the six-path tiles on a five-by-five grid.**

indicators (KPIs) so that participants may enhance their performance, minimize errors, and optimize their skills [14]. One of the most critical KPIs for a race car on a track is average lap time, as it directly reflects a vehicle's speed and efficiency [16].

We chose to modify this metric by using the time taken by the vehicle to complete a specific segment. To obtain such performance data, we can either have players conduct test runs on various racetracks or utilize AI agents to operate and race each car on these segments. We elected to employ AI controllers to simulate these tracks, due to the large number of segments we have generated.

Numerous research efforts have been devoted to creating self-driving vehicles. Examples include user-recorded data-induced car controllers [2], reinforcement learning-based car controllers [18][17], or genetic algorithm-based car controllers [20]. Any of these will work for our purposes, we have implemented AI agents programmed to follow a race line.

6.1 Racetrack Segment Completion Time

After selecting an appropriate AI controller, we proceeded to compute the completion times for each segment across two vehicles. These calculated values were then documented alongside corresponding track segment identifiers and vehicle types in preparation for the creation of a final racetrack compilation.

We started with two vehicles, Car A and Car B and simulated them on all racetrack segments. Generating diverse racetrack segments provides varied performance among vehicles. Analyzing the summary statistics presented in Table 1 for completion times on all segments, we observed that the standard deviations for completion time for Car A and Car B are 7.99 and 5.87 respectively, meaning Car A completes each segment within 28 to 44 seconds and Car B completes each segment within 23 to 34 seconds.

Upon further examination of the data, we see that Car B's mean value stands at 29.26, contrasting with Car A's higher mean value of 35.92. Moreover, 75% of Car B's completion times fall below 33.21, while this threshold is 41.37 for Car A. This indicates that Car B exhibits superior overall performance compared to Car A.

Car	Mean	STD	Min	Max	25%	50%	75%
A	35.92	7.99	5.00	70.20	30.69	36.07	41.37
B	29.26	5.87	13.09	64.88	25.51	29.55	33.21

Table 1: Summary statistics on all 24,918 segments for two possible vehicles.

To verify if there exists a sufficient number of segments where Car A outperforms Car B, a plot depicting the difference in completion time for each racetrack segment, see Figure 6, has been provided. The x-axis represents racetrack segment IDs, while the y-axis displays the completion time obtained by subtracting Car B's completion time from Car A's completion time. The red line signifies a value of 0, indicating equal completion times for both vehicles in that particular segment. Data points above this line indicate longer completion times for Car A, and vice versa if below the line. While Car B is clearly better for most segments, this figure demonstrates ample data entries on both sides of the line, with 24, 566 entries above the line and 352 entry below the line, allowing us to confidently propose the final racetrack assembly algorithm.

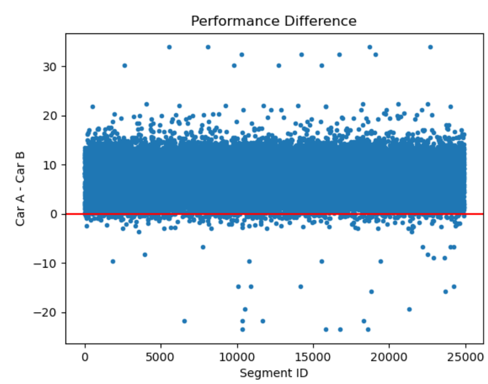

Figure 6: Completion time difference on each racetrack segment. X axis represents racetrack segment ID. Y axis represents completion time difference.

7 RACETRACK ASSEMBLY

Given the data in Section 6, we can systematically select and arrange the segments of the racetrack to form a closed circuit, ensuring that the overall design of the racetrack provides an equitable competitive environment for all categories of vehicles. Mathematically, let $T = \{t_1, t_2, \cdots, t_n\}$ be the set of all expanded racetrack segments. Let $V = \{v_1, v_2, \cdots, v_w\}$ be the set of all vehicles and let c_{ij} be the completion time for vehicle v_i on track t_j. Let $S = \{s_1, s_2, \cdots, s_m\}$ be sequence of integers where t_{s_i} denote an expanded racetrack segment. We want to create an S with the following constraints:

$$\{t_{s_1}, t_{s_2}, \cdots, t_{s_m}\}\text{forms a loop, and} \tag{1}$$

$$\sum_{k \in S} c_{1k} \approx \sum_{k \in S} c_{2k} \approx \cdots \approx \sum_{k \in S} c_{wk} \tag{2}$$

7.1 Selection Algorithm

To satisfy Constraint 1, we first randomly construct an S such that the selected expanded racetrack segment forms a loop. For constraint 2, we first calculate

$$\sum_{k \in S} c_{1k}, \sum_{k \in S} c_{2k}, \cdots, \sum_{k \in S} c_{mk}$$

respectively. We then calculate the average of these values as

$$a = \frac{\sum_{k \in S} c_{1k} + \sum_{k \in S} c_{2k} + \cdots + \sum_{k \in S} c_{mk}}{m}$$

Finally, we calculate the difference between each term and the average, square it, and sum it to form the mean squared error

$$error = (\sum_{k \in S} c_{1k} - a)^2 + (\sum_{k \in S} c_{2k} - a)^2 + \cdots + (\sum_{k \in S} c_{mk} - a)^2$$

The *error* denotes the imbalance within the racetrack; as this value approaches 0, the track becomes increasingly balanced. To minimize the error, a formal grammar approach is employed. Initially, we commence with a complete racetrack utilizing the generated segments, see Figure 7. We then calculate the *error*. Subsequently, we select a segment, annotated by the the red square. This selected segment is substituted with an alternative segment, numbered in black. If the new *error* value is less than the previous one, the substitution is accepted; otherwise, it is rejected. We then repeat the process until the *error* is below a given threshold, λ. Our substitution rules are straightforward, permitting each racetrack segment to be replaced with another segment that possesses identical opening configurations. The complete selection process is formalized in Algorithm 1.

Algorithm 1 Selection Algorithm

while $error > \lambda$ **do**
  Randomly select $s_{old} \in S$
  $s_{new} \leftarrow t_s$ where the vehicle fallen behind performs the best
  $e \leftarrow$ the new error value with the new segments
  if $e < error$ **then**
    $error \leftarrow e$
    $s_{old} \leftarrow s_{new}$
  end if
end while

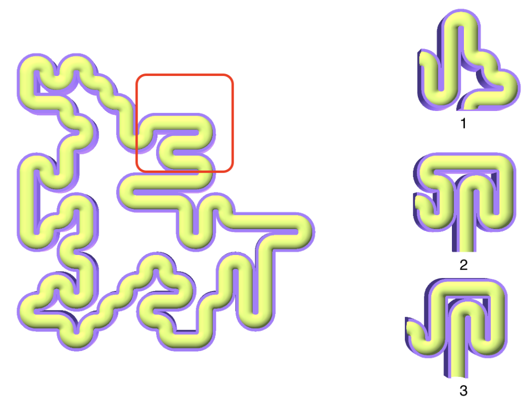

Figure 7: The provided image features an arbitrary racecourse, with a designated section enclosed within the red square being targeted for replacement. A series of alternative segments are presented alongside, identified by numerical black markers, which may serve as potential replacements for the selected segment.

8 EXPERIMENTAL RESULT

Our experimental outcomes encompass three parts. The first part highlights a concrete use case, while the second part focuses on demonstrating the algorithm utilizing bézier curve-based racetrack segments. The third part presents noteworthy discoveries that can serve as a foundation for future research endeavors.

8.1 Use Case

Initially, we designed a racetrack comprising 100 segments, see Figure 8. Each horizontal edge was formed by 40 linear straight

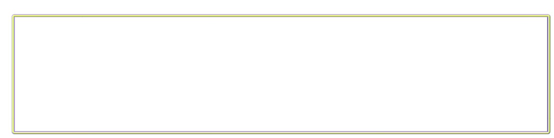

Figure 8: A racetrack constructed using 100 tiles.

segments, while the vertical edges were established using 8 such segments. The four corners features 4 turning segments each. We then applied the selection algorithm to only replace one-third of the straight segments until the track is fair and obtained the racetrack in Figure 9. Using the same set of segments, we can permutate and obtain

$$\begin{aligned} P(100, 32) &= \frac{100!}{(100-32)!} \\ &\approx 3.763099 \times 10^{61} \end{aligned}$$

different racetracks.

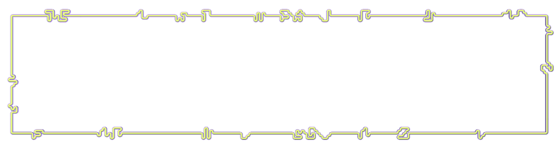

Figure 9: A fair racetrack obtained by applying the substitution algorithm to the racetrack on Figure 8.

8.2 Bézier Curve Segments

To showcase the flexibility of our algorithm, we incorporated track segments using Bézier curves. By experimenting with various combinations and configurations, we successfully produced diverse Bézier curve, see Figure 10. In this particular example, we utilized three points to generate the necessary segments. We ensured that the tangents at both endpoints remained fixed while randomly adjusting the position and tangent of the second point, resulting in the formation of straightaway segments. We repeated the process mentioned in Section 6 on a set of two different vehicles and recorded the performance data. Subsequently, various track layouts were created, with some segments designated for substitution. Our algorithm was then employed to generate fair racetracks, as showcased in Figure 11. The tracks on the left depict the initial layouts, the red squares mark the segments chosen for substitution. The tracks on the right represent the resulting fair racetrack configurations generated through our substitution algorithm, with two vehicles finishing the race within 3 seconds from each other.

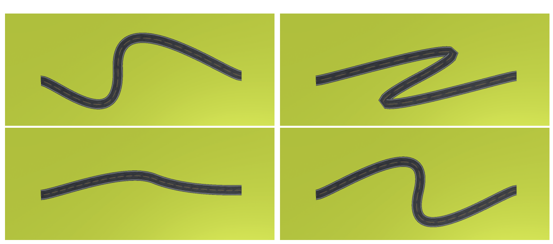

Figure 10: Some racetrack segments acquired using Bézier curves.

8.3 Discovery

After applying our algorithm to various racetracks, we have plotted the cumulative differences in completion times for both Car A and Car B as they traversed through each segment of the course. This analysis has revealed several intriguing insights.

Figure 12 shows one example. The racetrack is shown on the top. The x-axis on the plot represents racetrack segments that they visit in order, and y-axis displays the accumulative completion times obtained by subtracting Car B's accumulative completion time from Car A's accumulative completion time. For example, the value of y at $x = 2$ represents the completion time difference when both vehicles finished traveling the first two segments. If the value is positive, it indicates that Car A spends more time completing these

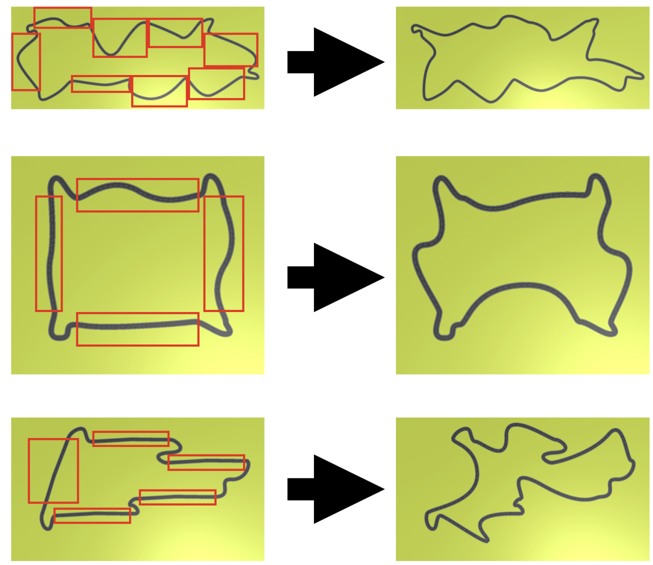

Figure 11: Unfair racetracks (Left) and fair racetracks (right).

segments (Car B is in lead). And vice versa (Car A is in lead) if the values is negative.

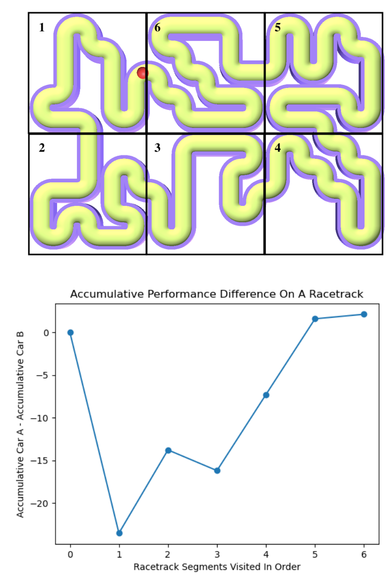

Figure 12: Fair racetrack along with performance difference plot. The red dot is the starting point of the track and track is oriented counter-clock wise. Track segments are numbered in black.

The plot demonstrates a trend where Car B initially falls behind, exhibiting an accumulative performance difference below zero upon completing the initial segment; however, as the race progress, it steadily narrows the gap between itself and Car A, ultimately converging towards zero accumulated difference.

In contrast to Figure 12, Figure 13 presents an alternative scenario where Car B assumes a leading position at the outset and toward the conclusion of the competition, with the accumulative difference being above 0 during segments 0 to 4, while Car A exhibits a remarkable surge near the end that results in a substantial reduction of the gap between both vehicles, as the accumulative difference falls below zero from segments 4 to 5.

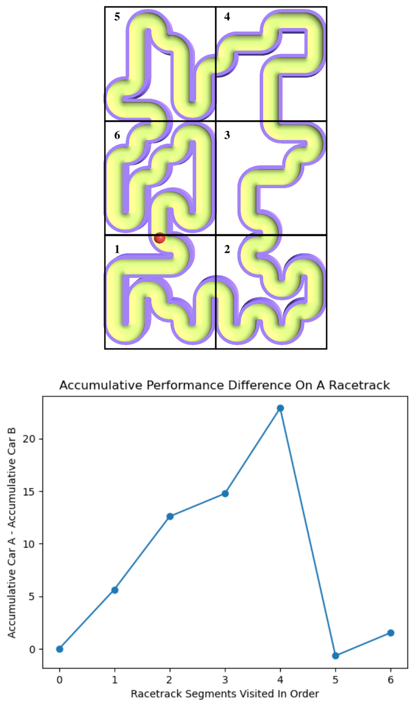

Figure 13: Fair racetrack along with performance difference plot. The red dot is the starting point of the track and track is oriented counter-clock wise. Track segments are numbered in black.

A different racetrack layout, see Figure 14, portrays another narrative where Cars A and B interchangeably close the distance between them throughout the race, leading to a more dynamic and oscillatory performance trajectory. The reason for such uniformity is that we are essential osculating two segments with different orientations. This drama can be controlled and future work will look into having the designer produce or difference graph and searching for a racetrack that provides that behavior.

9 FUTURE WORK

The proposed theoretical framework demonstrates versatility in accommodating a wide range of track segment attributes and AI-driven driving features. While our present examples may be somewhat restrictive due to exclusion of additional elements, like height variations, overhangs, and tunnels, incorporating these aspects into the segments can significantly enhance the intrigue of racetrack layout design.

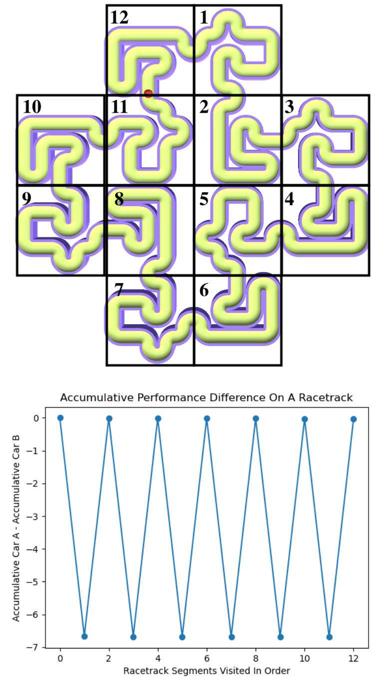

Figure 14: Fair racetrack along with performance difference plot. The red dot is the starting point of the track and track is oriented clock wise. Track segments are numbered in black.

For future research, investigating the impact of distinct vehicle controllers on simulation outcomes is recommended, particularly in relation to accommodating diverse human playing styles and assessing their influence on racetrack generation procedures.

Regarding error metrics, this investigation primarily focuses on ensuring racetrack fairness. As a potential area for further exploration, developing multilayered error formulations and optimizations could extend beyond fairness to encompass additional intriguing gameplay objectives.

Furthermore, the study introduces its framework using two vehicles; expanding this framework to include more vehicles and assessing the robustness of the framework is beneficial for future studies. Additionally, designing a framework that can produce different car stats such that each car has some advantage in certain segments is valuable.

Lastly, the previous section's findings raise an interesting question: given a designer sketch of accumulative time difference, how can we generate a racetrack that satisfies such conditions?

10 CONCLUSION

In this paper, we present a novel three-step framework aimed at mitigating racetrack biases. We explore various alternatives and enhancements for each step in our proposed methodology. The first stage involves generating racetrack segments using two distinct approaches - 5x5 tiling and Bézier curve. A crucial constraint is enforced, ensuring that entries and exits align seamlessly across these segments. The second step is to gather performance data using self-driving AI, we demonstrated this using a race line following AI. Finally, a grammar substitution algorithm is utilized to address racetrack biases by optimizing an error function specifically tailored for this purpose. We showed some possible future research that may focus on alternative methods or augmentations of each step within our proposed model. This innovative framework serves as a solid foundation upon which further developments can be built in the pursuit of reducing racetrack disparities.

REFERENCES

[1] Hafizh Adi Prasetya and Nur Maulidevi. 2016. Search -based Procedural Content Generation for Race Tracks in Video Games. *International Journal on Electrical Engineering and Informatics* 8 (12 2016). https://doi.org/10.15676/ijeei.2016.8.4.6

[2] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. 2018. ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst. arXiv:1812.03079 [cs.RO]

[3] Alexander Becker and Daniel Görlich. 2020. What is Game Balancing? - An Examination of Concepts. *ParadigmPlus* 1, 1 (Apr. 2020), 22–41. https://doi.org/10.55969/paradigmplus.v1n1a2

[4] Fabian Behrens and Ulrich Gohner. 2020. Procedural race track generation for domain randomization. (2020), 4 pages.

[5] Luigi Cardamone, Pier Luca Lanzi, and Daniele Loiacono. 2015. TrackGen: An interactive track generator for TORCS and Speed-Dreams. *Applied Soft Computing* 28 (03 2015), 550–558. https://doi.org/10.1016/j.asoc.2014.11.010

[6] Luigi Cardamone, Daniele Loiacono, and Pier Luca Lanzi. 2011. Interactive evolution for the procedural generation of tracks in a high-end racing game. *Genetic and Evolutionary Computation Conference, GECCO'11*, 395–402. https://doi.org/10.1145/2001576.2001631

[7] Jared E. Cechanowicz, Carl Gutwin, Scott Bateman, Regan Mandryk, and Ian Stavness. 2014. Improving player balancing in racing games. In *Proceedings of the First ACM SIGCHI Annual Symposium on Computer-Human Interaction in Play* (, Toronto, Ontario, Canada,) *(CHI PLAY '14)*. Association for Computing Machinery, New York, NY, USA, 47–56. https://doi.org/10.1145/2658537.2658701

[8] Linus Gisslén, Andy Eakins, Camilo Gordillo, Joakim Bergdahl, and Konrad Tollmar. 2021. Adversarial Reinforcement Learning for Procedural Content Generation. *CoRR* abs/2103.04847 (2021). arXiv:2103.04847 https://arxiv.org/abs/2103.04847

[9] Alexander Benjamin Jaffe. 2013. *Understanding Game Balance with Quantitative Methods*. PhD thesis. University of Washington. Available at https://digital.lib.washington.edu/researchworks/bitstream/handle/1773/22797/Jaffe_washington_0250E_11528.pdf?sequence=1&isAllowed=y.

[10] Quanyi Li, Zhenghao Peng, Qihang Zhang, Cong Qiu, Chunxiao Liu, and Bolei Zhou. 2020. Improving the Generalization of End-to-End Driving through Procedural Generation. *CoRR* abs/2012.13681 (2020). arXiv:2012.13681 https://arxiv.org/abs/2012.13681

[11] David Maung. 2016. *Tile-based Method for Procedural Content Generation*. Ph. D. Dissertation. The Ohio State University, The Ohio State University. http://rave.ohiolink.edu/etdc/view?acc_num=osu1461077485

[12] Luke McMillan. 2011. A Rational Approach To Racing Game Track Design. https://www.gamedeveloper.com/design/a-rational-approach-to-racing-game-track-design.

[13] Erik Jhones F. Nascimento, Tassiana M. Castro, Ana Carolina S. Abreu, Filipe A. Lira, and Amauri H. Souza. 2021. Procedural Generation of Isometric Racetracks Using Chain Code for Racing Games. In *2021 20th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. 136–143. https://doi.org/10.1109/SBGames54170.2021.00025

[14] NSCA's Essentials of Sport Science. 2023. Skill-Based Key Performance Indicators.

[15] Les Piegl and Wayne Tiller. 1996. *The NURBS Book* (second ed.). Springer-Verlag, New York, NY, USA.

[16] Alex Ryzhkov. 2023. Car Racing Track Core 7 KPI Metrics to Track and How to Calculate.

[17] Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. 2017. Deep Reinforcement Learning framework for Autonomous Driving. *Electronic Imaging* 29, 19 (Jan. 2017), 70–76. https://doi.org/10.2352/issn.2470-1173.2017.19.avm-023

[18] Yusef Savid, Reza Mahmoudi, Rytis Maskeliūnas, and Robertas Damaševičius. 2023. Simulated Autonomous Driving Using Reinforcement Learning: A Comparative Study on Unity's ML-Agents Framework. *Information* 14, 5 (2023). https://doi.org/10.3390/info14050290

[19] Noor Shaker, Julian Togelius, and Mark J. Nelson. 2016. *Procedural Content Generation in Games* (1st ed.). Springer Publishing Company, Incorporated.

[20] Yago Sáez, Diego Perez Liebana, Oscar Sanjuán, and Pedro Isasi. 2008. Driving Cars by Means of Genetic Algorithms. 1101–1110. https://doi.org/10.1007/978-3-540-87700-4_109

[21] Tim J. W. Tijs, Dirk Brokken, and Wijnand A. IJsselsteijn. 2008. Dynamic Game Balancing by Recognizing Affect. In *Fun and Games*, Panos Markopoulos, Boris de Ruyter, Wijnand IJsselsteijn, and Duncan Rowland (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 88–93.

[22] Julian Togelius, Renzo De Nardi, and Simon M. Lucas. 2007. Towards automatic personalised content creation for racing games. In *2007 IEEE Symposium on Computational Intelligence and Games*. 252–259. https://doi.org/10.1109/CIG.2007.368106

[23] Rodrigo Vicencio-Moreira, Regan L. Mandryk, and Carl Gutwin. 2015. Now You Can Compete With Anyone: Balancing Players of Different Skill Levels in a First-Person Shooter Game. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 2255–2264. https://doi.org/10.1145/2702123.2702242

[24] Mister Wu. 2023. Mario Kart 8 Deluxe in-game statistics. https://www.mariowiki.com/Mario_Kart_8_Deluxe_in-game_statistics.