# TORCS: The open racing car simulator

Bernhard Wymann* Christos Dimitrakakis†
Andrew Sumner† Eric Espié‡ Christophe Guionneau‡

March 12, 2015

## 1 Introduction

The open racing car simulator (TORCS [14]), is a modern, modular, highly-portable multi-player, multi-agent car simulator. Its high degree of modularity and portability render it ideal for artificial intelligence research. Indeed, a number of research-oriented competitions and papers have already appeared that make use of the TORCS engine. The purpose of this document is to introduce the structure of TORCS to the general artificial intelligence and machine learning community and explain how it is possible to tests agents on the platform.

TORCS can be used to develop artificially intelligent (AI) agents for a variety of problems. At the car level, new simulation modules can be developed, which include intelligent control systems for various car components. At the driver level, a low-level API gives detailed (but only partial) access to the simulation state. This could be used to develop anything from mid-level control systems to complex driving agents that find optimal racing lines, react successfully in unexpected situations and make good tactical race decisions. Finally, for researchers that like a challenge and are also interested in visual processing, a 3d projection interface is available.

## 2 The simulation engine

The engine uses a discrete-time simulation, with simple Euler integration of differential equations. The discretisation is set to $0.002s$ of simulated time. The simulator is geared towards simplicity, yet it handles all basic elements of a vehicular dynamics. This includes (i) the basic properties of the vehicular system such as the mass and rotational inertia of the car, engine, wheels and other components (ii) mechanical details such as different suspension types, links and differentials (iii) dynamic and static friction profile of tyres for different ground types, and finally (iv) a simple, but realistic, aerodynamic model including slipstreaming and ground effects.

*Project leader
†Project developer
‡Project founder

As TORCS is completely modular, the simulation can be easily replaced.[1] This would be of particular interest to researchers interested in developing low-level controllers for electronic drive systems.

# 3 The robots

In TORCS, the participating players are referred to as "robots". They are loaded as external modules in TORCS. This means that new artificially intelligent agents can be developed independently and they only have to satisfy the basic API requirements for robot code. At the moment, a large number of dedicated TORCS robots exist, some of which can operate at a level exceeding that of human performance in the game. Consequently, they form a challenging metric against which any new AI player can be evaluated.

Before every race, each robot can gather and process information about the track's geometry and surfaces. It is up to the user to decide how much of this information use. This is an opportunity to calculate a reasonable initial racing line for the track, perform a suitable set-up for the cars and decide upon a team and pit strategy.

The robots have the opportunity to interact with the simulation every $0.02s$. The default interface is through a low-level API[2] which can provide detailed information about the race status to the robot, exact position, distance from the edge of the track, the position of other cars, etc. However, there are many parts of the simulation state to which the robots have no direct access. Consequently, even the basic driving problem is *partially observable.*

Robots may also use a calculated three-dimensional projection, instead of the low-level API. This is intended to be used for researchers that have an interest in visual processing. However, the overall problem then becomes much harder as there is significantly less information directly available.

# 4 The racing problem

The racing problem could be split into a number of different components, including robust control of the vehicle, dynamic and static trajectory planning, car setup, inference and vision, tactical decisions (such as overtaking) and finally overall racing strategy. With only a single car on the track, the overall problem can be formalised as a partially observable Markov decision processes. However, in general it is a partially observable stochastic game. Having said that, there are a number of different challenges of varying difficulty that may be formulated within the context of racing.

The *trajectory planning* problem is the problem of finding an optimal trajectory according to some criterion (such as the time to complete a track). In the static case, we are given the track geometry and calculate the trajectory. The dynamic case involves calculating trajectories on the fly. This can happen for many reasons such as a lack of geometry information, or an unexpected deviation from the trajectory, or the appearance of obstacles.

[1] In fact, a more complex simulation with fewer simplifications in the contact forces and aerodynamics is also available.

[2] A thorough description of the API can be found at `http://www.berniw.org/aboutme/publications/api-1.3.6.tar.bz2`

If a trajectory has been pre-planned, a *robust control* problem would principally involve maintaining the desired trajectory, as well as more low-level problems such as making sure that the car is stable and tyres don't spin or lock out. At a higher-level, the *minimal lap time* problem is to find a driving policy that minimises the expected time taken to complete one or more laps of the track. It is easy to see such an objective can be formalised via an additive utility function, c.f. the racetrack problem in [11].

The *inference* and *vision* problems appear when the robot is using the 3d-projection interface rather than the detailed low-level API information. In that case, the robot must infer its speed, its position on the track, the relative location of other cars, the distance to obstacles and track-edges, etc. However, inference problems already exist even with the API, since the robots do not have access to the simulation itself. Consequently, they must have some level of uncertainty regarding the effect of any action they take.

The *overall* racing problem itself can be formalised as maximising the probability of winning a race, minimising expected race rank or maximising the expected number of points obtained for the team in a championship setting. While it is possible to formulate an additive utility function for this problem, it would necessarily be very sparse and so difficult to optimise. In that sense, it would be similar to the utility function in games such as go [8].

# 5 Conclusion

There is now a large set of software surrounding TORCS, such as an online interactive track generator [4]. In addition, there are now multiple forks. One of them is Speed Dreams [7] which is geared towards a better human player experience, pyTorcs [5], which is a port of TORCS to Python replacing many modules with standard open-source software.

There are also two major competitions using TORCS. One of them is the simulated car racing championship [10], organised around the evolutionary computation conference GECCO and the computational intelligence in games (CIG) confernece. This is mainly a research-oriented event. The second is the annual TORCS endurance world championship[13], which usually lasts around 6 months every calendar year. Due to its length, the event is geared towards hobbyists, but would also benefit from academic entries.

As a result, more than 300 research papers have now been written employing TORCS as a basis, primarily as a test-bed for artificial intelligence algorithms. While most of those use the low-level API features, or the auxiliary API for driver development [3], others are more ambitious and try to employ vision [9, 12]. However, TORCS has found other uses, beyond AI research. For example, it has been used to create an automotive test-bed [6], to perform a study on driver attention and stress [1], to develop highway platooning controllers and to access economical driving for trucks [2].

TORCS continues to evolve, and we feel that it has a lot more to offer to both hobbyists and the academic research community. One of our main goals is to maintain a stable API so as to avoid disruption for its many users. Finally, the modular architecture of TORCS and open source licensing allows it to continue to live long after its original creators and current maintainers have moved on.

## Acknowledgements

Many thanks go to all past contributors, and in particular to Rémi Coulom, Charalambos Alexopoulos and Andrew Sumner. Finally, we wish to acknowledge the efforts of Luigi Cardamone, Daniele Loiacono and Pier Luca Lanzi, who have turned TORCS into a highly successful competition platform CIG and GECCO.

## References

[1] Alexandre Benoit, Laurent Bonnaud, Alice Caplier, Phillipe Ngo, Lionel Lawson, Daniela G Trevisan, Vjekoslav Levacic, Céline Mancas, and Guillaume Chanel. Multimodal focus attention and stress detection and feedback in an augmented driver simulator. *Personal and Ubiquitous Computing*, 13(1):33–41, 2009.

[2] Tales Nereu Bogoni and Marcio Sarroglia Pinho. Use of a simulator to assess the application of economic driving techniques by truck drivers. In *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, pages 3020–3026. IEEE, 2012.

[3] Clara Caldeira, Claus Aranha, and Guilherme N Ramos. Torcs training interface: An auxiliary api for developing torcs drivers. 2011.

[4] Luigi Cardamone, Daniele Loiacono, and Pier Luca 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*, pages 395–402. ACM, 2011.

[5] Keith Curtis. pytorcs. `https://github.com/KeithCu/PyTorcs`, 2013.

[6] Utsav Drolia, Zhenyan Wang, Yash Pant, and Rahul Mangharam. Autoplug: an automotive test-bed for electronic controller unit testing and verification. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pages 1187–1192. IEEE, 2011.

[7] Andé Gaëtan, Beelitz Wolf-Dieter, Xavier Bertaux, Eckhard M. Jaeger, Kristóf Kály-Kullai, Gábor Kmetykó, Enrico Mattea, Haruna Say, Joe Thompson, and Simon Wood. Speed dreams v2.0. `http://www.speed-dreams.org`, 2013.

[8] Sylvain Gelly and Yizao Wang. Exploration exploitation in go: UCT for monte-carlo go, 2006.

[9] Jan Koutník, Giuseppe Cuccu, Jürgen Schmidhuber, and Faustino Gomez. Evolving large-scale neural networks for vision-based torcs. 2013.

[10] Daniele Loiacono, Luigi Cardamone, and Pier Luca Lanzi. *Simulated Car Racing Championship Competition Software Manual*, 2011.

[11] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. 1998.

[12] Cuong Tran and Mohan M Trivedi. Towards a vision-based system exploring 3d driver posture dynamics for driver assistance: Issues and possibilities. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 179–184. IEEE, 2010.

[13] Bernhard Wymann. The torcs endurance world championship. `http://www.berniw.org/trb/`, 2008–2013.

[14] Bernhard Wymann, Eric Espié, Christophe Guionneau, Christos Dimitrakakis, Rémi Coulom, and Andrew Sumner. TORCS, the open racing car simulator, v1.3.5. `http://www.torcs.org`, 2013.