# Design and Requirement

The goal of the project is to develop a virtual 3D racing environment where the difficulty of the opponent racer is dynamically adjusted. We implemented a time trial mode. The goal is to drive a lap as fast as possible. Players are challenged by a “transparent” opponent e.g. ghost car. In traditional race games the ghost car is a reproduction of your previous runs. We additional introduce the Virtual Rival (VR). VR is a ghost car adjusted on the players estimated skill level. The skill level is estimated from the players’ previous runs. The trajectory of the VR run is taken form a database of runs from all players. Virtual rivals teach and guide while entertain and challenge. Players can improve driving skills in every lap and VR improve with them.

# Architectural analysis

There are different stockholders involved in this project. The developers who want to create a challenging environment and track the players’ motivations, emotions and performances. The other stockholders are the players, who want to be entertained. Therefore, different requirements have to be met. The solution should be easy to implement and meet the requirements of both parties.

The following section outlines some basic requirements of developers and players which have to be satisfied. Since multiple players should be able to play at the same time, a suitable platform has to be identified. Developers should be able to integrate tools to measure the learning effect in different situations. The generated player data should be stored in a structured, reliable and everywhere accessible way. Finally, there are some requirements on the data analysis.

**Requirements regarding the Virtual Racing Environment**

The virtual racing environment is the race game where players are challenged by traditional ghosts and VRs. The virtual racing environment is very important for the learning experience. The structure and realism of the race influences the players’ motivation to play.

*Control:* To fit all player types the controls should be as simple as possible. This means players without gaming experience or driving experience should be able to play without difficulties.

*Graphics*: It is essential that the created race game is designed in a way that invites players. The objects should evoke associations with a real race track.

*Multiplayer*: The environment has to support multiple players because implementing such a feature is in the scope of this work. It should be possible for multiple users to complete the tracks at the same time.

*Maintainability*: Once created, developers must be able to identify software bugs and fix them. It should also be possible for programmers to extend the implemented functionality anytime.

**Requirements regarding the Questionnaires**

The questionnaires are the main tool to gather information about the players. They are essential to identify the players’ personality capture and the emotional reactions during the race.

*Integrated*: The questionnaires should not influence the game flow.

*Clear*: It is important to have clear and simple questions. The controls during the questions should be self-explanatory and fault resistant.

*Fast*: The loading times between questions should be fast. Time should be spent playing, not answering questions.

**Requirements regarding the Player Statistics**

Player statistics are used to identify player types and skills. The data is recoded in each lap for every player.

*Hidden*: The data recording should be in the background. The process must need only little resources.

*Extensibility*: After the development is finished, the developers should be able to add additional statistics.

**Requirements regarding the Data Storage**

The data storage handles all data created by the players. The data is needed for the method and player analysis.

*Reliable:* The data should be accessible at all time.

*Fast*: The loading and saving times should be as fast as possible to reduce latency.

**Requirements regarding the Statistical Evaluation**

Evaluation involves collecting and scrutinizing every data sample in a data structure where the analysis can be performed. Statistical evaluation is used to uncover trends and patterns within the data.

*Extensibility and Flexibility*: It should be possible to change metrics and add additional metrics.

*Fast*: The data evaluation should be reasonably fast even for large data sets.

# Architectural synthesis (Design)

For the virtual rival framework several modules will be developed and implemented to create a competition based race simulation which ensures motivation and enjoyment. The goal should be to improve while having fun. This section focuses on models that can be integrated in virtual driving environments.

Figure shows a simplified version of a potential scenario using a distributed approach to impart driving skills. Analysts can add new race tracks and modify the artificial intelligence of non-player characters. Accompanying concept questions before and after the races are used to assess the learning progress and find driving patterns. Players can access the Virtual Rival World whenever they want. There is no limit of players the world can handle at the same time. The players start by reading introductions and finishing a first tutorial drive. A first data analysis assesses their initial driving skill. After that, they try to improve their times in additional rounds. Questionnaires are used to track their motivation.

# Selecting a Game Engine

Game engines art tools that simplify the game development process. There various game engines with a different philosophies on game development and aiming at a wide range of different needs (Christopoulou):

* Engines that do not require programming
* Web-technology based engines
* Open-Source engines which are customizable
* Professional game engines

The game engine massive influences the game development process and potentially has technical limitations. Petridis introduced a framework for comparing engines based on six criteria’s:

* Audiovisual Fidelity: Rendering, Animation, Sound
* Functional Fidelity Scripting: Supported AI Techniques, Physics
* Composability: Import/ Export Content, Developer Toolkits
* Accessibility: Learning Curve, Documentation, Support, Licensing, Cost,
* Networking: Client Server/ Peer–to- peer
* Heterogeneity: Platform Support

This section compares the three most popular game engines (populareengines) with additional focus on our requirements[ref]:

* Unreal
  + Powerful
  + Games: Gears of War, Mass Effect
* Unity
  + One size fits all
  + Games: Overcooked, Superhot
* Game Maker
  + No programming needed
  + Games: Super Crate Box, Undertale

**Audiovisual fidelity**

Table shows an overview of the audio visual features. Game Maker supports only sound. Unity and Unreal engines support all graphics technologies. These features are important to provide a realistic environment for the player.

Table 1: Audovisual Fidelity based on Petris and Christopoulou

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | Game Maker | Unreal | Unity |
| Texturing | x |  |  |
| Lightning | x |  |  |
| Shadows | x |  |  |
| Special Effects | x |  |  |
| Animation |  |  |  |
| Sound |  |  |  |

Table summarises the result regarding functional fidelity. All the nominated game engines provide support for various AI techniques and scripting.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Game Maker |  |  |
| Script |  |  |  |
| Collision Detection |  |  |  |
| Path Finding |  |  |  |
| Decision Making | x |  |  |
| Basic Physics |  |  |  |
| Rigid Body |  |  |  |
| Vehicle Dynamics | x |  |  |

**Composability**

It is important that the engine provides features to import from common data sources (see Table). Game Maker is the only engine which does not support importing/exporting from the best known CAD platforms (Blender, 3D Studio Max, MAYA). Unity and Game Maker require a separate installation of Developer Toolkits while Unreal includes the required toolkits automatically.

**Accessibility**

Table shows Unity seems to be the most usable game engine, providing the most free tutorials, examples and assets, while its community is very large (Christopoulou). Unreal Engine is the only engine which provides free technical support. Game Maker is designed to allow users to easily develop computer games without having to learn a complex programming language (Game Maker).

**Networking and Heterogeneity**

The last step in the selection process is to weight the heterogeneity of the engines and their network support (see Table). All the nominated game engines have a similar network model and support multiplayer games. Unity has an important advantage in game development since it supports all platforms (especially consoles and mobile).

**Conclusion**

Based on our analysis of three leading game engines we can draw some important conclusions regarding the effectiveness of each game engine for our race simulation.

*Game Maker* is not suited for 3D games. Therefore, it can’t be used for a realistic race simulation. However, it is ideal for people without programming experience.

*Unity* and *Unreal* are the most powerful game engines, which support almost all of the features needed for a race simulation.

*Unreal* supports all features and is completely open source.

*Unity* is the best all-round engine. The main reason for choosing Unity over Unreal for our race simulation is the big asset store, which allows us to focus on developing. While the Unreal Marketplace has grows tremendous, the Unity Asset store is still the industry frontrunner (UnityAsset).

Game Engines Selection Framework for High-Fidelity Serious Applications - Petridis

Overview and Comparative Analysis of Game Engines for Desktop and Mobile Devices – Christopoulou

<https://www.gamedesigning.org/career/video-game-engines/> - populareengines

<https://en.wikibooks.org/wiki/Game_Maker_Programming> (Game Maker)

# <https://arstechnica.com/gaming/2018/07/epic-ups-unreal-marketplace-creators-pay-well-above-industry-standard/> - UnityAsset

# Conceptual Architecture

For the *Virtual Rival Framework* several software modules will be developed and implemented to create a scenario which ensures that competitive driving can take place. The goal should be an acquisition of player driving data in a stimulating environment. The design of the *Virtual Rival Framework* itself will be the subject of chapter (ref), whereas the current chapter is going to describe the functionality of the developed modules based on the requirements imposed by the concepts of motivation, emotion and performance in competition and the players. Figure illustrates the modular structure of the *Virtual Rival Framework*.

**Driving Module**

The *Driving Module* is the main component of the framework. The Driving Module incorporates the main car mechanics: car controls, gear shift, exhaust sound, data capture.

* Car controls: Vehicle drivetrain capable of providing torque to two or four wheels. Accelerating, Steering and Braking based on player input.
* Gear shift: Automatic gear shift similar to real life automobile gear shift systems.
* Engine: Characterises the engine speed and exhaust sound.
* Data capture: Records important driving metrics e.g. acceleration, speed and position.

**Skill Adjustment Module**

The Skill Adjustment Module has two functions. Primarily, adjusting the skill level of players based on race results. Secondly, estimate the initial skill level of new players.

**Player Enjoyment Assessment Module**

The *Player Enjoyment Assessment* Module measures the emotions and motivation of the players at all times.

**Data Management Module**

The *Data Module* extends the *Virtual Rival Framework* and stores player information.

**Data Analytics Module**

The Data Analytics Module accesses player information through the Data Management Module. The data is used to measure trends and analyse player performance.

**Visual Analytics Module**

The *Visual Analytics Module* accesses player information through the *Data Management Module*. It exemplifies data using graphs, maps and diagrams.

**Schematic process**

Figure gives an overview of how a typical racing experience in the virtual rival environment with the help of the implemented tools could look like.

First, players will have complete initial questionnaires and personality test where we can know the player. Subsequent, players have time to learn the controls when completing the tutorial levels.

After they have completed the tutorial levels, they have to compete on race track against ghost opponents. At the end, the performances evaluation of the current players lap follows, to update the estimated skill level and adjust the opponent.

# Summary

Before developing the first prototype, different requirements have been defined. First of all, general design aspects like availability, performance, scalability, and extensibility have to be considered. In addition, the players’ motivations, emotions and performances should be tracked. For this purpose, a closer look at the important psychological aspects was taken. The different focuses of players and analysts regarding a scientific racing environment were also taken into account.

It meets all the requirements, especially in terms of network model, support and accessibility. Unity already includes many tools with the free standard installation. Some components for the Virtual Rival Framework have to be implemented from scratch:

* Driving Module
* Skill Adjustment Module
* Player Enjoyment Assessment Module
* Data Management Module
* Data Analytics Module
* Visual Analytics Module

These modules will include various functionalities which allow the analysts to define the race properties and track performance. The players produce vital data while exploring the race tracks.

A typical virtual rival round-trip was identified to consist of six steps:

1. Perform initial questionnaires
2. Exploring the race controls in the tutorial levels
3. Estimate initial skill level
4. Drive lap against ghost competition
5. Update skill level based on driving performances
6. Go back to (4) until the end of the challenge

The next chapter will deal more specifically with the tools developed, namely the Skill Adjustment Module, the Player Enjoyment Assessment Module and the Data Analytics Module

The final result of the *Virtual Rival Framework* will be the topic of chapter \ref.