Fury Road

A game of cars, soccer and flying stuff

Book of specifications

Deloche Tristan
Goetz Simon
Kirszenberg Alexandre
Olivier Cyril

January 15, 2016
The French Lions, Ltd. $\boxed{\mathcal{TFL}}$

Table of contents

1 Introduction

Fury Road is a mix between a car game and a soccer game. Inspired by the likes of *Rocket League* and *Burnout: Paradise*, we want to create our own variation of the genre. When it comes to car games, as soon as you leave the beaten path of the racing genre, there's still a lot of uncharted territory. With our debut on the video game scene, *Fury Road*, we intend on filling that gap.

A game of *Fury Road* is set in an arena where two or more teams are facing off. In the main game mode, not unlike soccer, the teams have to fight for the possession of a ball, while at the same time defending their own goal. After a given duration, whichever team scored the most goal in the enemy goal wins the game. While simplistic, this set of rules allows for the most freedom mechanics-wise. Other game modes might include playing a variant of volleyball or a vehicle deathmatch.

We intend to take full advantage of the powerful 3D features of Unity in order to kickstart a simple but functional game, with a focus set on mechanics and playability. While the underlying concept is simple, the execution is what matters most. The game will heavily rely on realistic physics simulation, occurring in real time across a multiplicity of clients. How well we've come to balance and tweak our set of features will directly influence the feel of the game, much more so than simply how much content we've managed to fit in it.

Hitherto, none of us has ever done a project the scale of *Fury Road*. It is pretty new to us, although some of us have worked on big projects in the past. So it will obviously bring us a lot of knowledge and teach us new ways of programming. Mixing a car game and a soccer game involves a lot of physics: the ball and its bounciness, the cars and their collisions with walls or even other cars, a camera that should follow the car without moving through walls.

Those aspects are not to take lightly. A project this big will also strengthen our friendship: working together for a whole semester can either be a bonding experience, or a death trap for friendship. Having already spent a semester together, we are confident in the robustness of our group.

But that is not all: deadlines, presentations, book of specifications... Three

of us have faced such odds before. One might think that it becomes easier with time, when it really doesn't. A game is a huge undertaking. There is always room for improvement, features, gameplay tweaks, designs, storylines, etc. But oftentimes, when you begin to add something new, it opens the road to new possibilities. It is up to you to decide when to stop and when to begin something new, whilst keeping the deadline in mind.

In short, this project is sure to bring us a lot of knowledge, discipline, strictness and cohesion.

2 Presenting the team

2.1 Name of the team

The French Lions was created the 8th of November 2015 by a group of 4 friends, who met at EPITA on the very first day of school.

The name of the group emerged as evident to us, after a long-running private joke. At first, there was Simon's hair, that looked much like a lion's mane. Obviously, we proceeded to call him The Lion. A few weeks later, Alexandre, in a heated debate, confidently asserted that lions were indigenous to France a long time ago. A couple of hours of googling and fact-checking later, Alexandre admitted that his belief was nonsensical.

This long argument stood in memories, and ultimately lead to the name we know and love, **The French Lions**.

2.2 Members of the team

- **Alexandre Kirszenberg (Team Leader):**
  Alexandre is the oldest lion. He is the alpha male of the tribe. With the absence of females in the tribe, he has to settle for Simon. After discovering programming at the age of 15, he's made it his duty to write the best software ever conceived. When he's not delusional, he likes to play video games, read books and watch movies.

- **Tristan Deloche:**
  Tristan is the independant lion ; rival in ideals with the alpha one, yet respectful of the pack. Years of programming in non-academic fields unraveled to him the pleasure of taking on big projects through the architectural point of view rather than the algorithmic one. He is expecting a lot of interesting work with this one and is eager to put his experience and dedication at the service of the team (in exchange of their eternal indebtment, naturally). You could say he is like a mercenary.

- **Simon Goetz:**
  Simon, is a 16 years old boy coming from Strasbourg. He skipped two grades. He is the youngest lion and has a huge 4 years difference with the other members. He is creative and is the one that has wasted the longer time playing video games, so he can bring to The French Lions ideas for beautiful design of cars and arenas, interface and the menu, and also gameplay.

- **Cyril Olivier:**
  Cyril is 20 years old, originating from Paris. He studied in France then information technology and programming at the Swiss Federal Polytechnic School of Lausanne. He is the normal lion, intelligent and strong. With programming his passion, he readies himself to face his greatest challenge: dethroning the alpha lion...hum, ok later. Until then, bidding his time he roars towards all and prepares to help to the best of his abilities, the incredible journey ahead.

3 In-depth explanation of concept and gameplay

Dreaming of a good car game? Or maybe a good soccer game? Then look no further, because we plan to mix them together in order to design the most creative, elegant, yet deceptively simple game ever: *Fury Road*. While some play soccer normally, we asked ourselves: How could soccer be even better? We came to one conclusion: Add cars.

The concept in and out itself is quite elementary. Keep the idea of a field, two goals and a score, as well as pretty much all the rest. The only thing we are adding is the movement restriction – or freedom – imparted by the replacing of humans by cars. But it needs not end here! Let's add boost: Players will be able to greatly increase the speed of their car limited amount of time.

While this approach might seem rather odd to some, it holds great potential. Putting some restriction on movement greatly increases the difficulty of the game at first. However, after a while, the apparent complexity reveals its appeal. As the players' skills grow, so do the pleasure and the satisfaction they'll reap from the game.

At its core, the game will feature two parts: the menu, and the match. The menu will be welcoming us when we launch the game. Outside of games, it'll be the place where the player spends most of its time. From there, they'll be able to get into matches, change settings, customize their car, consult the leaderboards, and plenty more.

The match part is the real interesting part. This is where the gameplay will truly shine. Matches will be 5 minutes long, and we mean real minutes. Any interruption time will not be taken into account by the clock. Players' cars start the game at random predefined starting positions, where at least one player of each team will be able to get to the in the same amount of time. The clock starts when the ball is first touched, and the match goes on until a goal is scored or the timer runs out with equal scores on either side. Whenever a goal is scored, the game is temporarily stopped, the score of the scoring team is incremented, and cars are put back to random starting positions in time for

the kick off.

Regarding the physics in game, cars will be able to move with the same exact maximum speed, without boost of course. If a car bumps another one, both cars will bounce back proportionally to how hard they were hit at. This will add the possibility of displacing adversaries to move someone playing goalie for example. The ball has normal physics, and can be moved proportionally to the speed of the car that hits it. If the ball is "pinched", meaning it bounces quickly between two objects (car and car or car and wall), it will be propelled at huge speeds.

When the timer runs out, the team with the higher score wins the match. If both teams have the same score at the end of the imparted time, the game moves to overtime. In overtime, the first team to score will be crowned the winner.

In order to stop players from boosting all of the time, they will only have a limited amount of boost. A gauge on the HUD will instruct the player on their remaining amount of boost. This limitation brings a new dimension to the game, where players must use their resources with consideration, lest they miss a goal opportunity. Strategically positioned canisters will allow the players to refill their boost gauge.

The number of players in a match can vary. The plan is to have solo (1v1), doubles (2v2) and maybe even triples (3v3). Since the game relies mostly on multiplayer, there won't be much to do in single player. A training arena will allow the player to practice alone with unlimited boost. The player will be able to play against a basic AI, to practice shoots with penalty shootouts. Regarding matches in general, a single mode might be a limiter to the game. Different game modes with new rules will bring variety to the game. For instance, a VolleyBall mode, where the ball must not touch the ground, or even a map without goals, only springboard, and unlimited boost. Eventually, a car game wouldn't be a real one, without car's customization! A lobby where you can choose the color of wheels or the chassis and cool car's designs! The player will be able to choose between different arenas, and finally weather and time as rain or night.

4 Technological overview

*Fury Road*will be developed on the Unity game engine. The Unity game engine was created specifically to allow fast prototyping and iteration without sacrificing quality. Its ease of use and myriad of features – namely 3D graphics [1], Physics [2], Networking [3] and others – make it a top choice for the development of a game project. Furthermore, Unity works cross-platform.

While Unity offers a powerful and in-depth editor [], its main interest shines through its scripting capabilities. Absolutely essential in any game, Unity scripting will be the perfect way to precisely control the physical behaviour of the cars, create graphical effects and maybe even implement an AI system that goes further than just constantly moving towards the ball.

All scripting[4] will be written in C#, a strongly typed and secure programming language developed by Microsoft. Its powerful object-oriented paradigm will enable us to build lasting system-level components. Since Unity itself was written in C#, and given our brand new knowledge of the language, it seems by far the most obvious choice.

Fury Road will also heavily rely on physics[2]. Physics will dictate how the ball and other objects will interact with each other in game. As the default physics engine in Unity, as well as an industry standard, we hope that Nvidia PhysX will satisfy our needs. It is a powerful engine capable of real-time physics calculations on both the CPU and the GPU. More than 150 Triple-A games and tens of thousands developers use it. Among other cases, it will control the way the ball will move in the air or after a collision with a car, or the way cars will bounce back after hitting each other. Without it, we would have to ship our own simplistic physics engine. With a bit of luck and talent, the game might be playable, but not at all realistic and probably not enjoyable. Even though our idea is odd, the resulting game should feel right.

But even though the road seems traced in front of us, we need to be careful with how we use the physics engine. Since our game is quite unique, we can not expect everything we need to be included in the default library of the physics engine. After research, we found out that while Unity features its own collision handler for car physics, it appears to have a few flaws. As such,

we might have to create our own wheel simulation in order to best fit our needs.

It goes without saying that a good multiplayer game should support playing against remote adversaries. As such, we'll need to work with the networking part of Unity[3]. As with any physics-reliant game, synchronizing simulation across a wide range of client brings its fair share of issues. While it is possible, and perhaps even viable to use a peer-to-peer system, the absence of a centralized server to verify the physics calculation would be a huge limiting factor. Such a server could also be used to store player information and rankings. We will make up our mind along the way and decide which of the two options works best in our case.

And finally, our game will also need interfaces[5], such as the main menu or the HUD in game. Unity features its own UI system, that we intend to use to its fullest extent: animating our on-screen elements will bring life to our interfaces, and Auto Layout will allow us to target a wide range of screen sizes.

5 Economical overview & Timetables

Regarding the economic side of the project, us being students still living at the expense of our parents, we do not expect it to cost a lot to develop. Both the game engine and physics engine are free. Sharing our code and advancement through a private Github repository will not be a problem since Github graciously offers students benefits that would otherwise cost money.

The two only expenses the project might lead to will be in regard to the server we will use. Its purpose is to host the website, a SQL database to store information about our players, as well as your Unity game server. Since we might need a server with a bit of computing power, it could come to cost us 10€ per month to rent a mighty beast. In the case we greatly overestimated our needs, Amazon Web Services offer students credit to use on their platform. We'll also need to buy a domain name so as not to have to input a raw IP address to access the website.

Table 5.1: Task distribution

	Tristan	***Simon***	***Alexandre***	***Cyril***
Gameplay	**X**	**X**		
Physics			**X**	**X**
Graphics		**X**		**X**
UI	**X**	**X**		
Multiplayer	**X**		**X**	
Sound design		**X**	**X**	
Website		**X**		**X**
Player Rankings	**X**			**X**
AI			**X**	**X**
Game modes	**X**		**X**	

Table 5.2: Task completion

	First submission	***Second submission***	***Third submission***
Gameplay	**20%**	**60%**	**100%**
Physics	**10%**	**50%**	**100%**
Graphics	**10%**	**60%**	**100%**
UI	**10%**	**42%**	**100%**
Multiplayer	**5%**	**60%**	**100%**
Sound design	**0%**	**50%**	**100%**
Website	**40%**	**80%**	**100%**
Player Rankings	**0%**	**20%**	**100%**
AI	**0%**	**40%**	**100%**
Game modes	**33%**	**66%**	**100%**

Possible improvements in the case where the team would be ahead on time would be reactivity as well as quality of the graphics, further server-client speed improvements, anti-cheat system, localizations and a public API for modding purposes for example.

6 Conclusion

Such an undertaking should not be taken lightly. As a wise man once said, "Deadline is coming". As such, we fully expect to have to make choices and compromises in order to deliver not necessarily the best, but the most functional version of our game. We have many ideas regarding what our game should be, but not all of them will make it to the final product. On the other hand, as we move through production, new paths and issues will appear in front of us. Whether good or bad, we can't possibly expect to anticipate all of them. It will prove a challenge to overtake these unexpected occurrences, but we will learn a great deal of valuable lessons along the way.

7 Bibliography and references

Unity's reference manual for 3D game development:

1. Graphics http://docs.unity3d.com/Manual/Graphics.html
2. Physics http://docs.unity3d.com/Manual/PhysicsSection.html
3. Unet http://docs.unity3d.com/Manual/UNet.html
4. Scripting http://docs.unity3d.com/Manual/ScriptingSection.html
5. UI http://docs.unity3d.com/Manual/UISystem.html

Other links we used for research purposes:

1. http://www.nvidia.com/object/physx_faq.html
2. http://www.digitaltrends.com/gaming/how-to-make-a-video-game/
3. http://programmers.stackexchange.com/questions/125712/for-what-reasons-should-i-choose-c-over-java-and-c