
# Post Mortem

Fabrício Guedes Faria
Gabriel Arantes
João Victor Almeida
Rafael Pedrosa Clerici
Willian Gonzaga Leodegario

# Intro

This Post Mortem is a document in which we discuss a bit about how our project, *"No Prey, No Pay"* was made, what we were thinking, what went right and any sort of things we would have liked (if possible) to improve.

The game was made during a game development class at the University of São Paulo, during the second semester of 2018. We had roughly three months to make the whole game from scratch, as it needed to be exposed at USP Game Link, an event which happened at November 9th of 2018.

At the time, when the team was formed, most of us had known each other for a long time, which facilitated most of the brainstorming process. The group was comprised by people of diverse backgrounds and academic formation, a fact that contributed to the birth of a unique gaming experience. Our programmers, Fabrício, João Victor and Gabriel, all had different levels of programming skill, albeit having contact with game production before. João, in particular, had previous experience in the game industry, working at Ubisoft as an intern. Fabrício, member of the extracurricular group Fellowship of the Game (FoG), made games before, such as MulTI Task. Gabriel, also member of said group, having worked as an artist in Allpunk, worked for the first time in programming.

The art was almost exclusively handled by the Übermensch Willian Leodegario, with some small contributions by Gabriel. Willian worked tirelessly to provide an extensive collection of assets to be used in game. He is also associated with most of the extended goals we were able to achieve, all thanks to his utmost dedication to the project. Oh, and he also is a member of FoG.

In the design department, leading our group and making sure that everything was being kept together even through the hardest times, was Rafael Pedrosa. Having worked with game design before at FoG (I think we're onto a pattern here), he provided insight at many crucial points of our development process. Another important aspect was his interest in making the game feel as nice as possible, testing the game several times to ensure that controls, gameplay and game feedback were on point.

The very idea of our game, in fact, stems from a role-playing campaign DM'd by Rafael. The team got together to make a brainstorming session, pitching several different concepts for the project. One in a particular, making a fast-paced party game, loosely based on works such as Towerfall and Nidhogg, caught the general affection of the team. Rafael then came up with the idea of using lore, characters and world ambiance from his RPG adventure, and everybody loved it.

Our game was named "*No Prey, No Pay*", a reference to an old pirate motto. It means that a member of a pirate crew would only get paid based on whatever they could plunder. The name perfectly fits the central idea and lore of our game, and served as an extra "hyping" element to excite the development team. If you're reading this, I assume you have played the game before. If, however, you're an post-mortem afficionado or stumbled upon this document by chance, "*No Prey, No Pay*" takes place in a fantasy steampunk world, where a group of Kobolds, tiny furry and/or scaly creatures, have decided that piracy is the best way of living. Long story short, they recently acquired a hefty amount of loot, and will fight day and night until it's decided who will get the lion's share of the gold.

Looking at this description, it is easy to see our first worry: conveying everything to the player. Although steampunk ideas have recently been heavily ingrained in pop-culture, this was not the case for Kobolds. It was imperative that the art was effective in presenting these little creatures

not only as interesting beings, but also as bloodthirsty pirates. Moreover, their personalities needed to be well perceived even without any textual description.

Another foreseeable problem was the complexity of the game's code. It was clear to us that the game would require some good software engineering and code planning. Considering this, we worked on a class diagram, that was promptly abandoned by the team after a certain point (more details ahead).

The final challenge was to balance the whole thing: making sure that the controllers were tight, responsive. That the visual and audio feedback conveyed correctly what was happening. And, above all, that the game was fun. From the beginning, we knew that a good parcel of our time would be spent in this task.

It is important to state that the teachers of the discipline were paramount to the success of our final product. Acting as a "producer" of sorts, they provided the team with tips and information regarding the development progress, and were always easily accessible. The fact that they demanded a strict calendar of planning and shipping also aided the whole process. We first focused in producing the game design documents, and spent as much time as possible doing this. This allowed for a well defined project, giving confidence to everyone that we sort of knew what we had to do and deliver.

Later in the course, we had to pitch our game. The whole task of producing a playable demo, and making that a presentation that would sell our game made us consider problems that we didn't thought before, such as the score system. Besides that, the idea of selling your game as product really helps in making you believe that your work is good and worth it, really. Even if you have nobody to listen to your pitch, do it.

And finally, USP Game Link made us keep a strict development schedule, making sure we would be able to deliver everything we promised. We used the platform HackN'Plan to keep tabs on every task, using a sprint based model of work. Every week, we would present everything we'd done, and plan the thing we would do for the next week. We would also prepare for periods where the game development needed to be halted, due to graduation exams.

Anyway, we've talked a lot already. The next sessions of this post-mortem provide some insight on things we believe we did right, and things we believe that should've been done better (WAY better). I hope that whoever reads this can avoid some of the mistakes made during this project, and take full advantage of the things we did right.

# Things that went well

## 1. Team integrity

We had a discord server dedicated towards discussing and exchanging ideas about the game, which helped to keep everyone informed and up to date to whatever was going on. We also had weekly meetings in which we could talk about how things were and what should be done. All of us were excited to work on this project, as we are all fans of this kind of party game, and the overall aesthetic of it resonated with us all. That led to great productivity and no missed deadlines.

## 2. Content production

Although we had only one dedicated artist, he managed to keep producing a lot of high quality content throughout all the development of our project. This acted as a positive feedback loop, as his excitement with the project drove him to work harder, and the quality of his art made the rest of the team even more compelled to make a better game.

One thing that greatly helped was using bones to animate our characters. At first, we feared the quality would not live up to our expectations, but it proved to be more than enough and enabled us to quickly edit and create new animations for our characters, which would not be possible if we used frame-by-frame animation. It also enabled us to add new characters quickly, as we could use animations previously made with minimal tweaks.

Finally, as the music was designed in the final steps of the production by an external engineer, it was well fitted to the overall experience hence the main graphism was already available, and the sound designer had a good material to start with.

## 3. Prototyping

The prototyping phase started since the first week of planning, alongside the One-Sheet and Ten-Pager writing process. It was indeed essential to feel the earlier ideas of the game experience, and led to a more efficient programming pipeline in the later steps, hence it gave a north to the main idea of the system architecture as a whole, and the attended gestalt.

Furthermore, this early prototype was a good source of feedbacks and main possible issues in the gameplay, and gave the team more maturity when starting the actual prototype, that led to a consistent gold version.

Even though it is risky to prototype that early in the process, this was a wise choice that was successful thanks to the simplicity of the gameplay and the wide range of similar games in the market, in a locomotion gameplay perspective.

## 4. Fast paced action

Not only we achieved the desired *fast paced action* experience in the game, but simply choosing it was wise as it matched with all the requirements of the genre: time limited matches with high player engagement.

Achieving it was not trivial. This was a combination of a precise combat system, a smooth locomotion and handcrafted animation and music that followed our gestalt's guideline, created through an iterative development process.

## 5. Overall satisfaction with the game

When we exhibited our project at the USP Game Link, we managed to capture the attention of a lot of visitors. There were always at least four people playing our game, and we even set up another computer, enabling up to eight people to play simultaneously in two different matches, and many would keep playing match after match. Everyone enjoyed it, and barring minor bugs, no major problems were detected, either in gameplay or balance.

Not only did the visitors like it, but we as developers also loved how the game turned out. We even discussed about continuing development and adding new content!

# Things that could be better

## 1. Development complexity

Although our scope was accurately defined and our gameplay was not very innovative, we did not manage our code scalability very well. Many solutions were redesigned during the development process, such as the physics system, which was, for a while, handcrafted. Later, we made a mix with Unity's and ours, and such changes in major systems like these consumed some of our working hours.

Many systems were duplicated and the overall software architecture was lowly cohese, and this kind of issue could have been easily avoided with more subsystem specific classes/entities, in the same way as Unity does for its physics, audio and other engine subsystems.

## 2. Player confusion

In multiple times our players didn't know what was possible to shoot or hit in up direction: basically, the only way them figured it out was when one of us used this feature. The same thing almost happened with the stomp. Also, on the final stage of the project, we start to ask about the viability of this feature, but we keep it on the final version.

The only "out of ammo feedback" is when the player try to shoot and fail, that detail caused a lot of confusion during the gameplay, mainly because shoot is one of the main and most used mechanics in the game.

## 3. Weird bugs

As we did not manage the code complexity, it led to many unsolved bugs, and ones that we simply did not understand. A character could get stuck in the scenery as easily as it could spin forever after a dash. This sort of bug take out the player of the experience and usually causes unjust results in matches, disencouraging players.

These were small problems that could have been solved with a better software architecture and more bug solving dedicated time, since most part of them are due to code implemented during crunch time.

## 4. Lack of communication

Sometimes, our designated artist would simple make more stuff than we had planned for, which led to lots of art needing fixes. Things like our map, which was still being tested and tweaked, would need many fixes in order to match the game itself.

However, the programming and design team also made some blunders, like the defunct ledge grab mechanic. At first, we planned that characters could grab the edges of platforms, but we did not have a clear idea on how that was going to work. In the end, this mechanic was implemented, albeit in a buggy way which would not blend well with the rest of the game and was scrapped, wasting some good work time.

## 5. “Crunch time”

Although not as severe as it could be, our team did experience some “Crunch time”, also known as “working really hard for long hours because deadlines are getting closer”. Our artist kept working everytime he could, which would hinder his ability to focus and work on the other classes he had to attend. The rest of us also had some unslept nights on our backs, and made our sacrifices to make the game as good as it could be. We believe the game only got truly fun after tweaks made the morning of the Game Linke event!

# Our final remarks

Even though our final game had its MVP with some additional features, it was clear that our lack of project management methods resulted in unexpected and unnecessary crunch time. The complexity of the project was totally under control during the whole development process, all the actors in the team were engaged in the development, and with some better planning, we could have achieved a more stable version of this game.

Furthermore, the lack of communication led the programmers to mislead the original game architecture, a huge mistake that we paid with a highly accoplated game system that became a nightmare in our final bug correction phase, and costed each of us time and patience.

Fortunately, our biggest asset was the team's passion and engagement to the project, that made us surpass these mistakes and deliver a simple and straightforward game, in a higher quality than our initial expectations. Each member worked as a gear in a big engine: resilient, reliable and in synchrony, putting its efforts in the right direction to achieve our final goal.