How Zjetpack was done

We came up with the task of developing a game. When we think of developing a game, we immediately start thinking about those games we played when we were kids, the good ol' days. Of course it had to be a game all three played. So we pointed to an old mobile game, Jetpack Joyride. That game was so much fun back then, but we thought we could do it better.

We first started developing a simple way to generate those enemies and implement basic physics for our main character, applying gravity to it, and of course a render capable of showing our entities. Should we generate those enemies randomly? Or should we do it predefined? That was one of our initial doubts. We ended up not doing it randomly for two reasons. Firstly, it's computationally complex and unpredictable, and secondly, there wouldn't be any feeling of levels and progressions. As a result, our levels are well plannified and calculated to fit with their respective levels and our player's learning.

We made our little engine to generate things and the simple physics to control our player, but, who is our player and who are those enemies? We needed to design them. That is where Luis Jesus made his well-done designs with their animated sprites. Now we had a player, some obstacles, an animation for them, and an engine capable of generating the enemies and moving everything around. This is a simple version of our game.

Yes, this simple version of our game doesn't have collisions, and it's too simple to be fun. But we had our code implemented to control all the entities we generated, a system to control what entities are alive and which ones are dead and should be removed, and a way to apply a component's functionality to an individual entity. From now on, making our game more complete and developing and applying new components to our engine will be easier.

The next step we made was implementing the collisions to our game. This is a very important component for the games because without collisions there's no obstacle to evade. This collision system will only need to check the collisions between the player and the generated enemies. So we did it. We implemented this little but important system. Our game was getting better.

From this point, the next implementations will be to complete our game and make it more fun. And with fun, monotone things aren't very associated, and right now our enemies have monotonous behaviors. We needed to implement a simple AI to make our enemies capable of changing their behavior and making them more challenging, so we created different simple behaviors for our enemies.

That 's good! We now have a fun and interactive game. But we have no levels or a menu to choose different actions. If we want to have different levels, we must also have a way to generate those levels at their appropriate time. This sounds like modifications on the previous things… Yes, but not much. We had to adapt the generator to be able to generate different combinations of entities, with a time lapse between them. Now our generator has "a list" of what to generate, and a counter to know when to generate the next entity. Yey!, one useful thing done.

We can now define different levels as different combinations of entities generated. Of course, the higher the level is, the harder it will be to survive against that combination of enemies generated, so we implemented a nice component that will control the level on which the player is, and it'll also tell the generator what combination he'll have to follow depending on the level. We, of course, needed a way to tell the player that he's starting a level. This is where we introduce an "intermediate level" between each level, on which we'll generate a simple entity that will say the level on which we are on in a nice animated way.

Last step is our precious menu. This required a lot of drawing and graphic designs. The code was easy to implement. Just a few keyboard checks, a code flow that shows the different menu options and start the game and we got our game menu done.

Oh wait! We forgot something, the music! Yes, we brought our musical side and composed our masterpiece for our menu and gameplay. We really hope you liked the music. The more you listen to the music, the more you want to listen to it. It 's addictive.

Final fixes will be adding life to our player, adding an option after death to restart the last level or go back to main menu, adding a power up that makes the player heal one life point, adding death counter, and after completing all levels, a little animation to show the player escaping.

And this is how our nice and simple game, Zjetpack, was done.