Level is a puzzle game. The player act on the environment to move a ball, collect tokens and advance through 3 different levels.

The player can use 3 different skills: move, jump and attack. The victory condition is to collect all tokens and/or defeat enemies and then access the exit. The failure condition is either losing all hps or falling off ground.

Three different levels are currently available. The first level aims to teach the movement mechanic. The second level aims to teach the jump mechanic. The third level introduce the health and the attack mechanic with a small mini-boss at the end.

The game mechanics are simple as I wanted to focus on the level design to provide the player a unique experience. The game can still be improved with nearly infinite possibility of level creation.

In this post, I will explain the creation process of the game. How the idea came, what I wanted to achieve and what challenges I faced.

History:

The creation of this game started two years ago. I am extremely interested in Level Design. I studied it through games and books and I remember being amazed by a simple concept. It came from the online book “The How’s and Whys of Level Design” by Sjoerd “Hourence” de Jong:

The utilisation of computers is not intuitive to humans. We must learn how to use it.

I came to realisation that it was true: all games, no matter what they are, have a tutorial to introduce the player to the game mechanics. All informatics tools have their own tutorial, documentation, etc…

I wanted to know if it was possible to create a game intuitive enough that someone can play it naturally without being frustrated (I consider that games like “Mario Bros” achieved this objective).

This was the beginning of the design of “Level”.

Design (Set Goals, Anticipate Challenges):

As a beginner Level Designer, I was clueless on how to design a game. I decided to create some precepts that I will have to follow:

* Stay simple: I wanted the game to be intuitive. That anyone can beat it without me teaching them the mechanics.
* Force the player to use the all mechanics: It would be a wasted to introduce a new mechanic and neve use it again/don’t use it at all.

Both were my main headlines when designing the game.

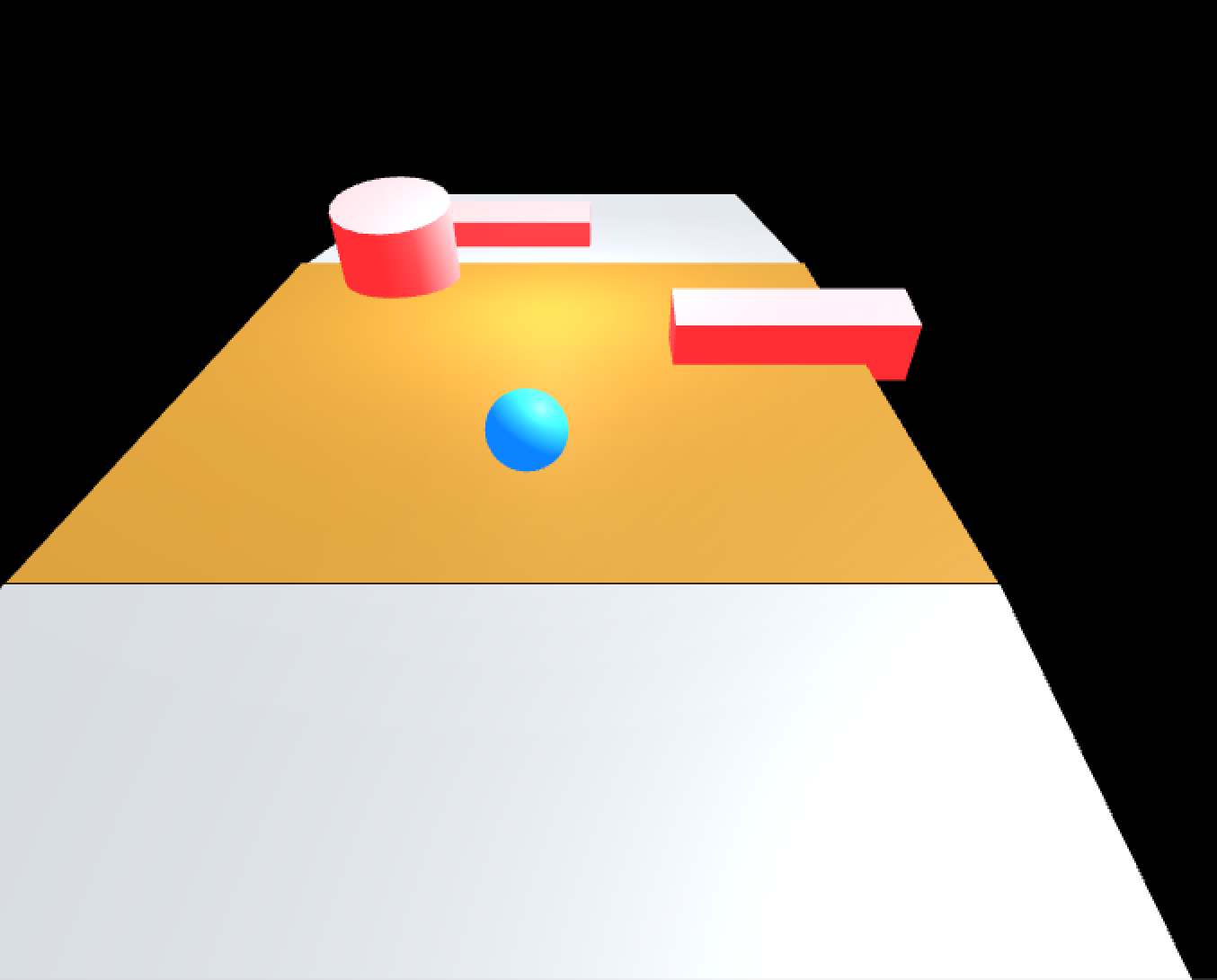
Then I decided the main mechanic of the game: move the ball by tilting the ground. It will allow the ball to move thought the world. It is the most important mechanic of the game, its core.

Design (Early-Stage):

I was unfamiliar with the way unity manages physics. I only worked with tutorials and 2D projects before.

I then imagined the behaviour of the ball relative to the ground I was aiming for. I wanted to ball to be easily movable. That if the player made a mistake, he/she can recover controls easily. Therefore, I decided the ball to have a small initial acceleration and a late terminal velocity.

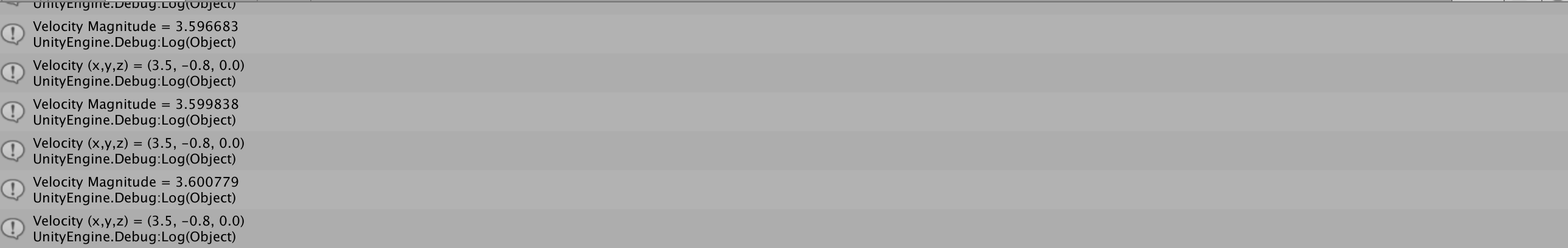
It was time to experiment with the ball’s Rigidbody and gravity.



I came to design my first prototype: a ball in free fall without any commands.

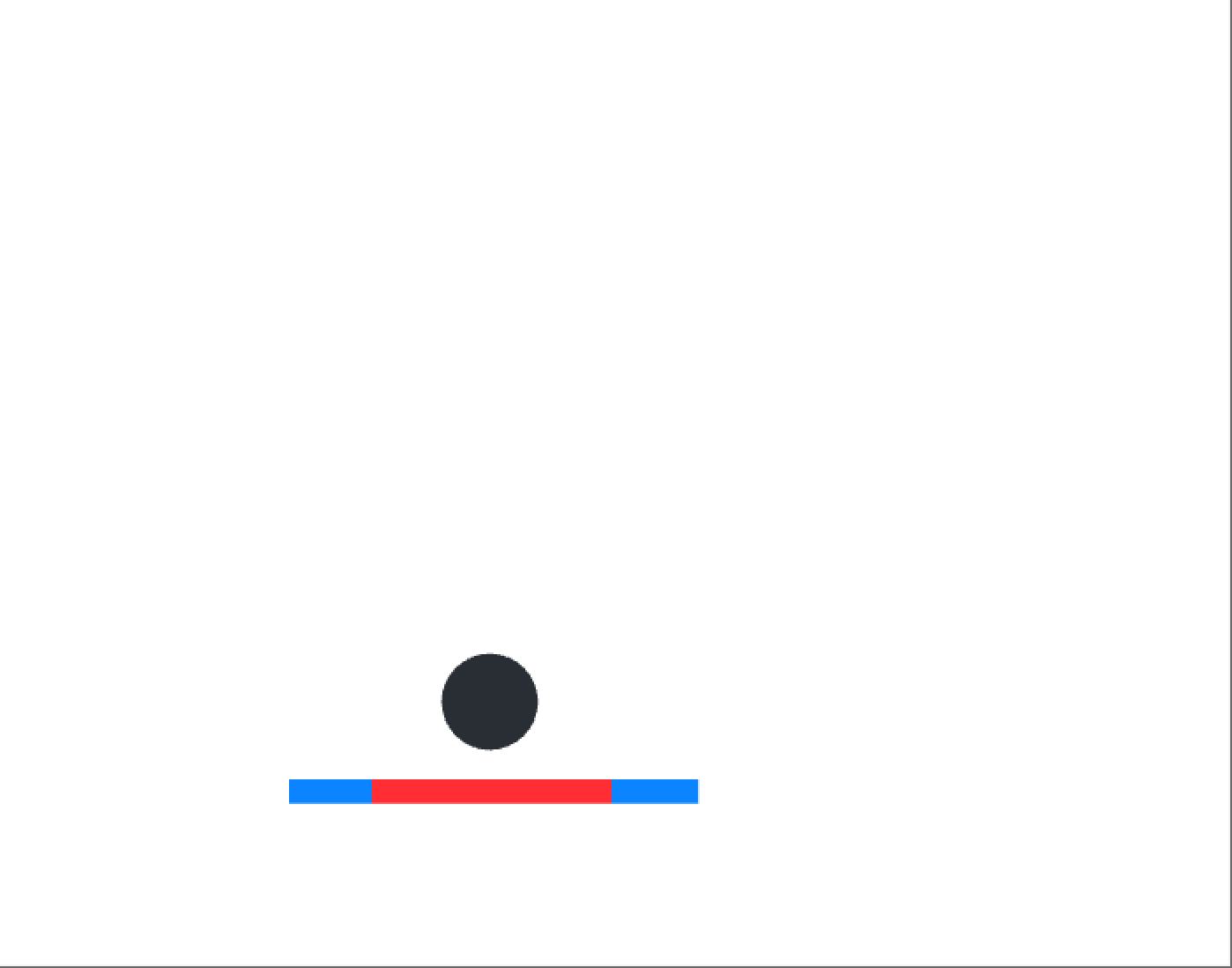
I wanted to understand how the velocity component worked, it’s proprieties, and how we can impact that with collisions.

I played around with the weight of the ball and the gravity to find a good ball acceleration relative to an angle. I was aiming the ball to get 4 velocity (magnitude) from a resting position on a plane angle of 10 degrees before 5sec. Why these values? because I needed a reference for the game. I wanted to have a “feel” of it and then adjust on what I thought would be good.



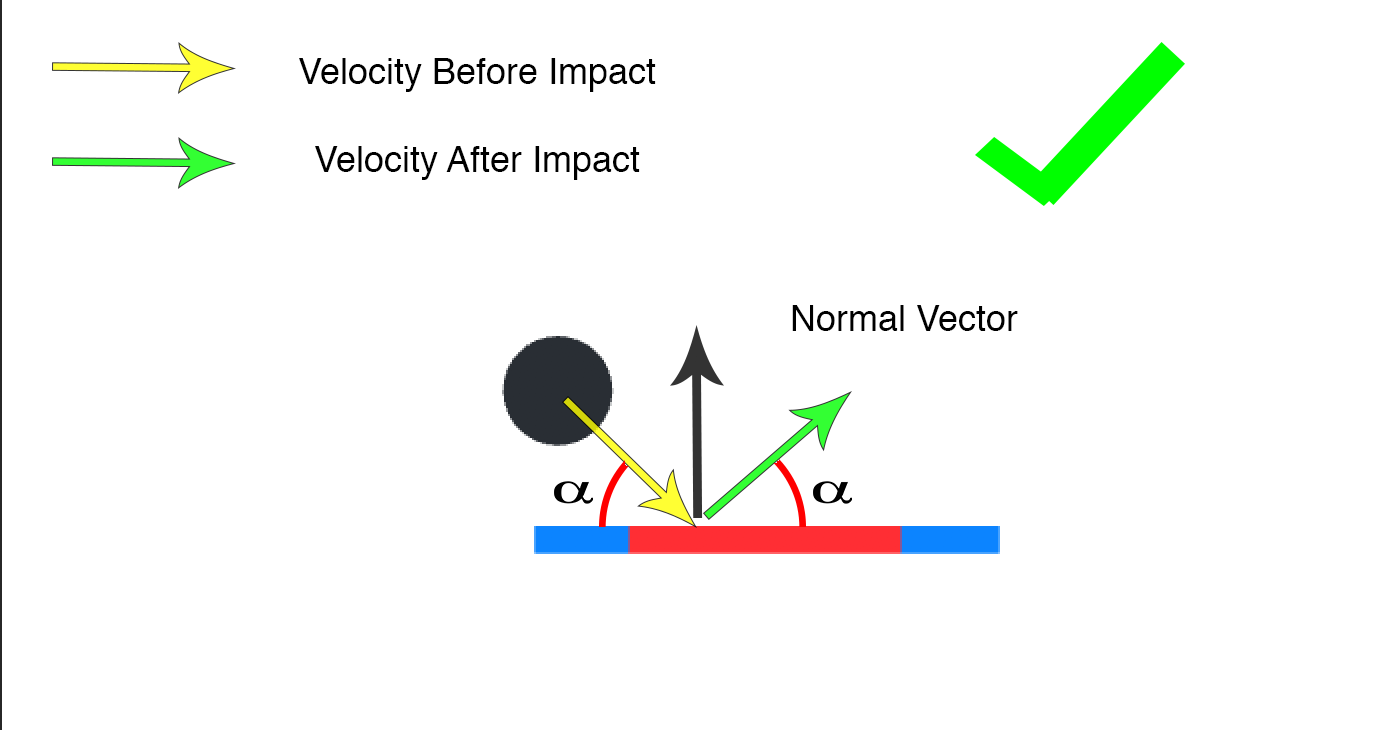
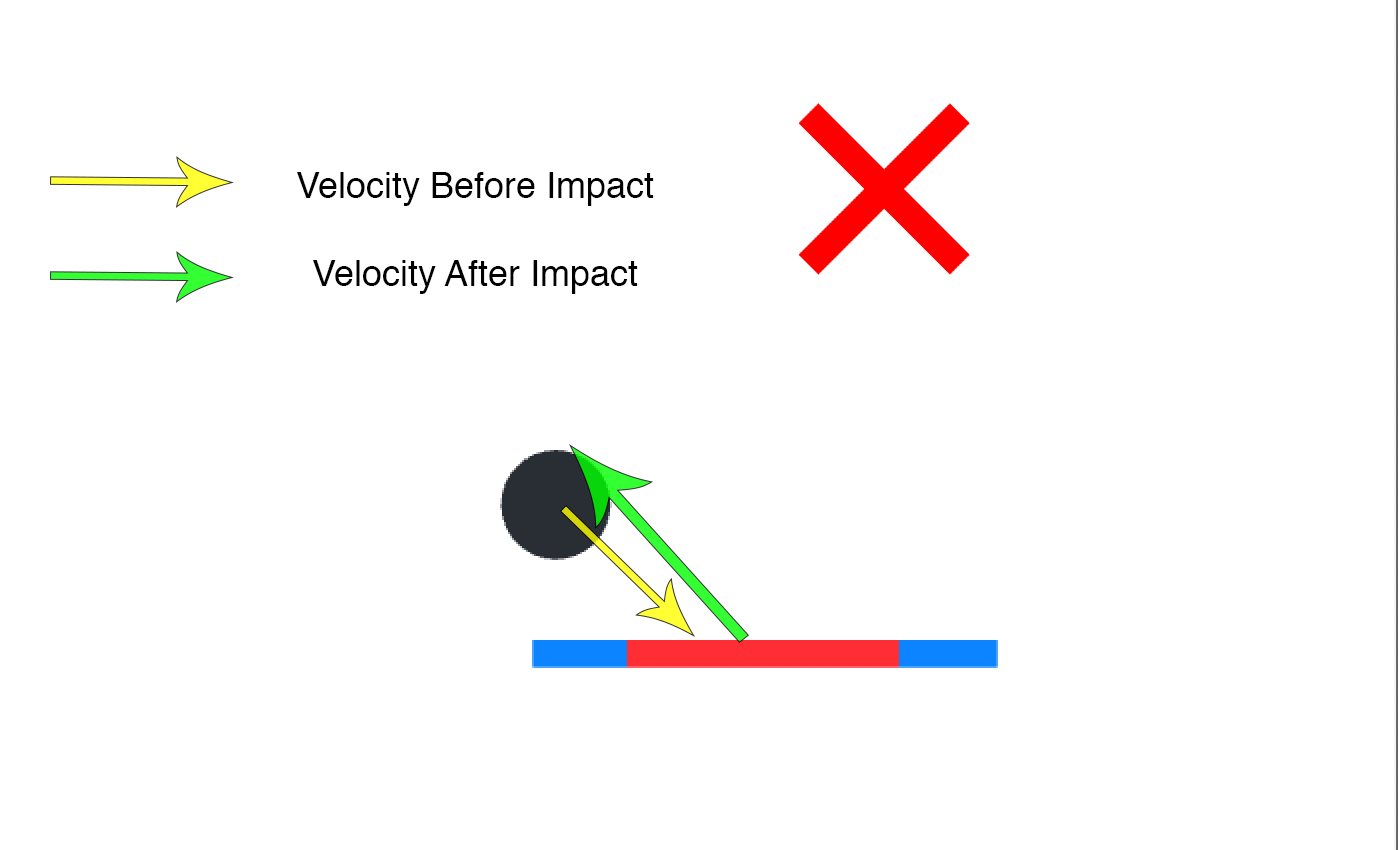
At this stage of the prototype, I realised that once the ball collides with a wall it stops in the wall and doesn’t move afterwards. I understood that bouncing back from the wall wasn’t a physic from the Rigidbody (or the material?). I explored a way to make the ball bounce with another prototype.

I did experiment with a 2nd prototype.



The solution I found (and that satisfied me) was to calculate the velocity before impact, reverse it and add it to the ball with the AddForce function. It was nice but not optimal as this method didn’t return the angles.

I had to perform vector calculations to find the new angle:



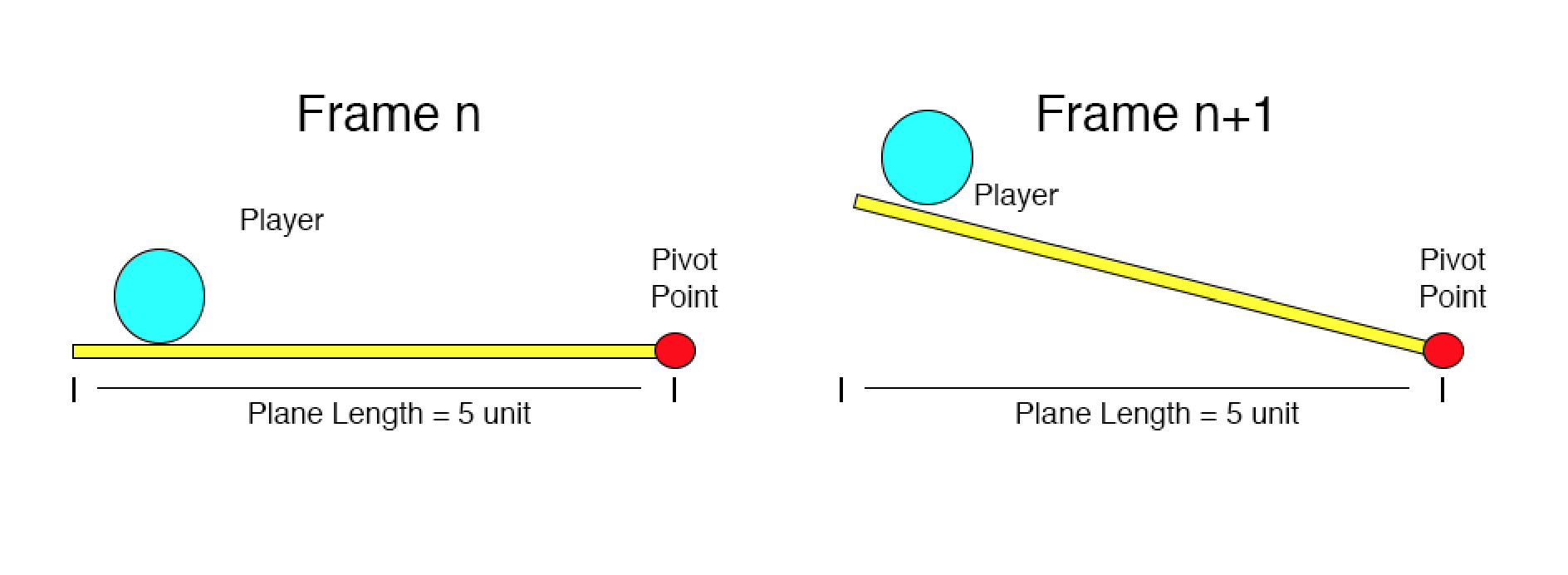
Once the ground and walls were fully operational. The behaviour of the ball seemed “natural”. I then started creating the game.

Design (Early-Stage: Hardest Challenge to solve)

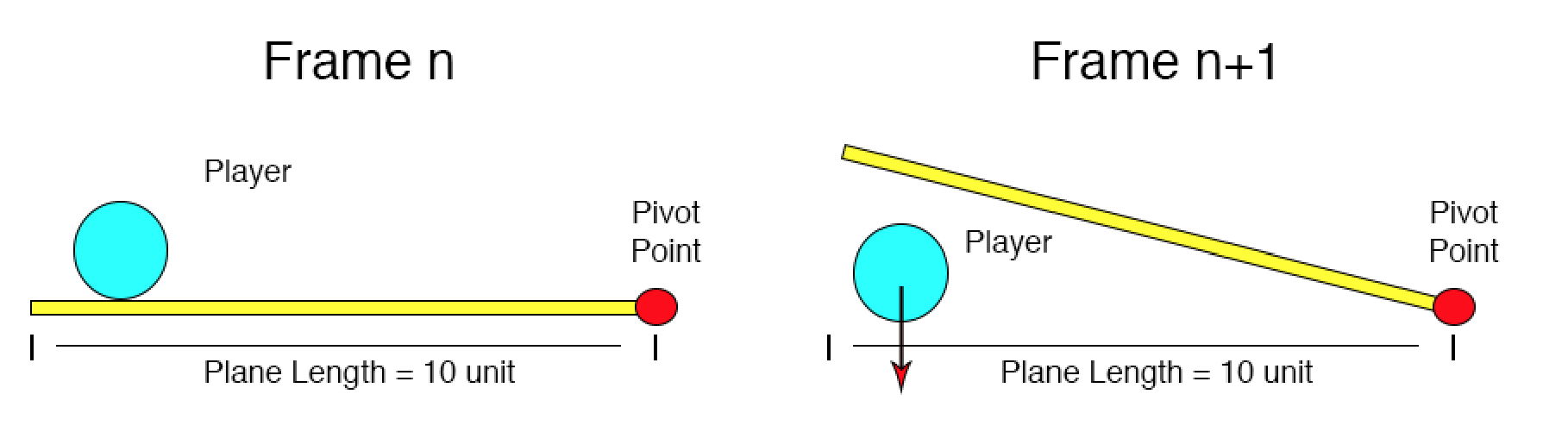
One of the hardest challenges I had to solve is the problem linked to the plane moving.

Because the Update() function update itself every frame, the vertical displacement of the plane, made with a Quaternion.Slerp (lerp from a point a to b by a displacement t), is dependant of the SIZE of planes.

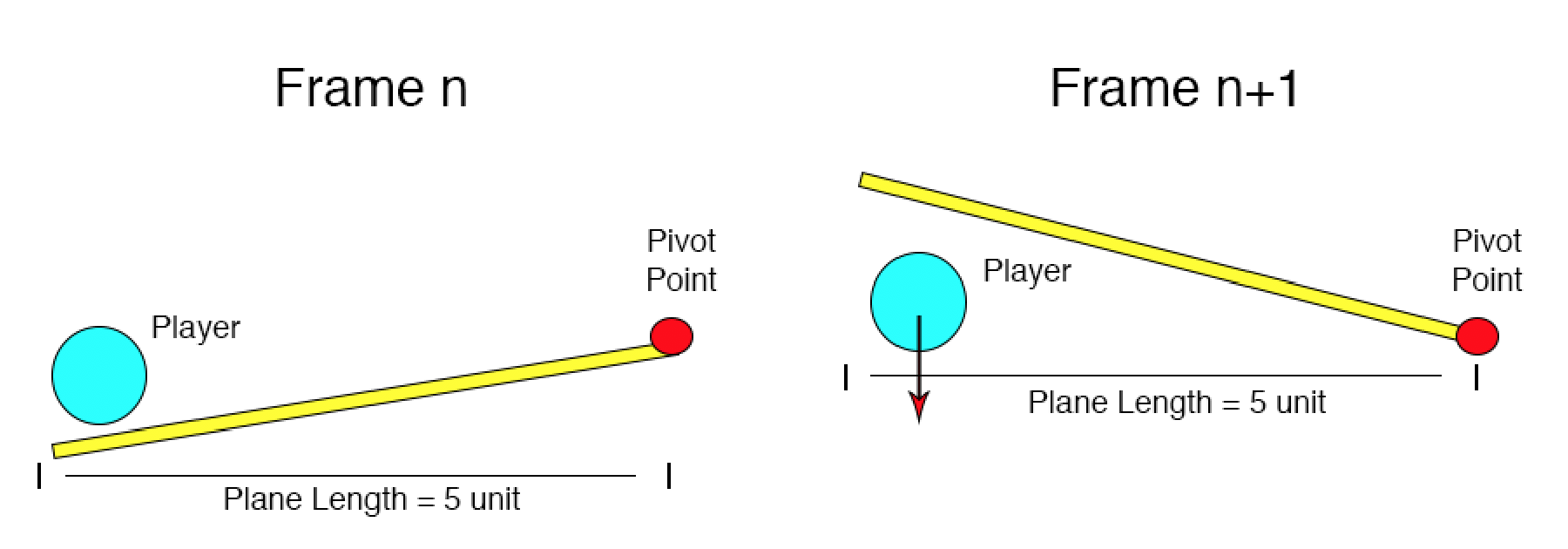
Let me explain: if the displacement of the plane per frame is less than the size of the ball’s collider, the ball will follow the plane such as bellow:



If the plane is too big, the displacement per frame increases. If it becomes bigger than the ball collider, the ball doesn’t follow the plane and falls.



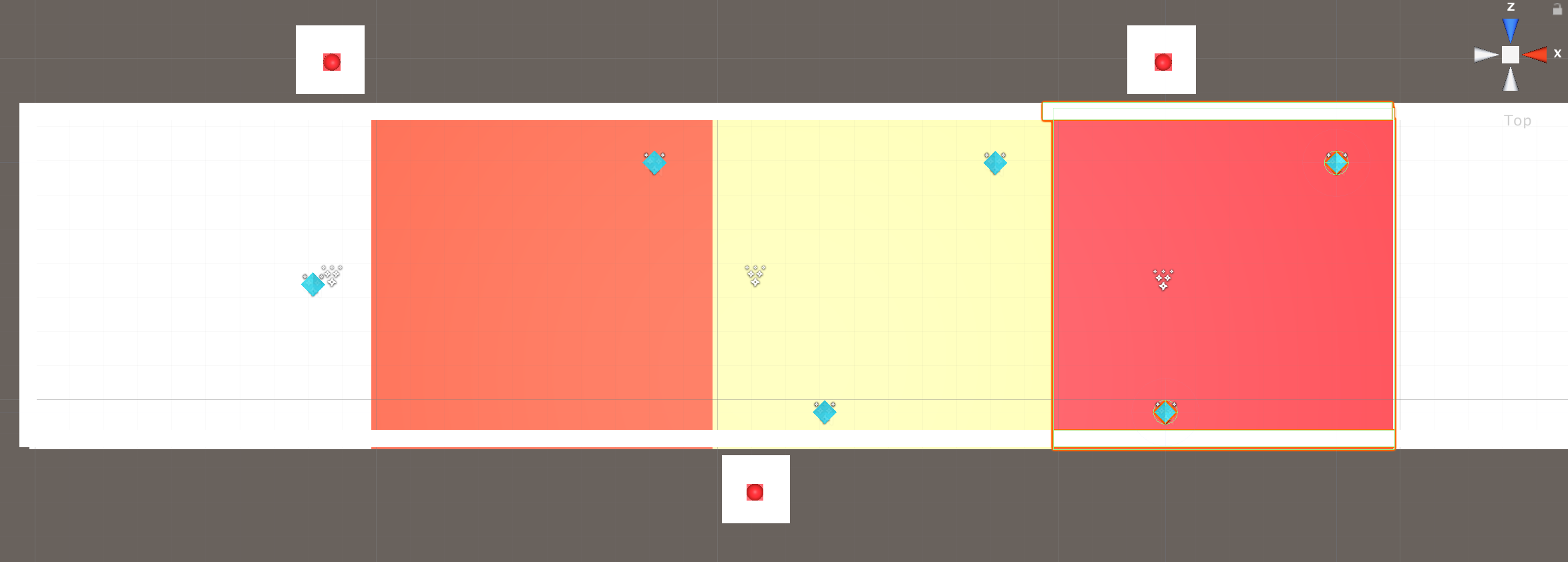
It becomes really tricky when the Player goes close to an end and then quickly switch between tilting left and tilting right. The displacement becomes even bigger:



Same applies when you jump really close to a plane’s edge. The plane will reset to Quaternion identity as the ball will exit the plane’s collider. The player will most likely keep the button pushed down, resulting to a similar situation. The player will sometimes fall off ground.

To solve this, I had no choice but to limit the plane length, limit the maximum rotation of planes and increase the refresh rate of the physics calculation.

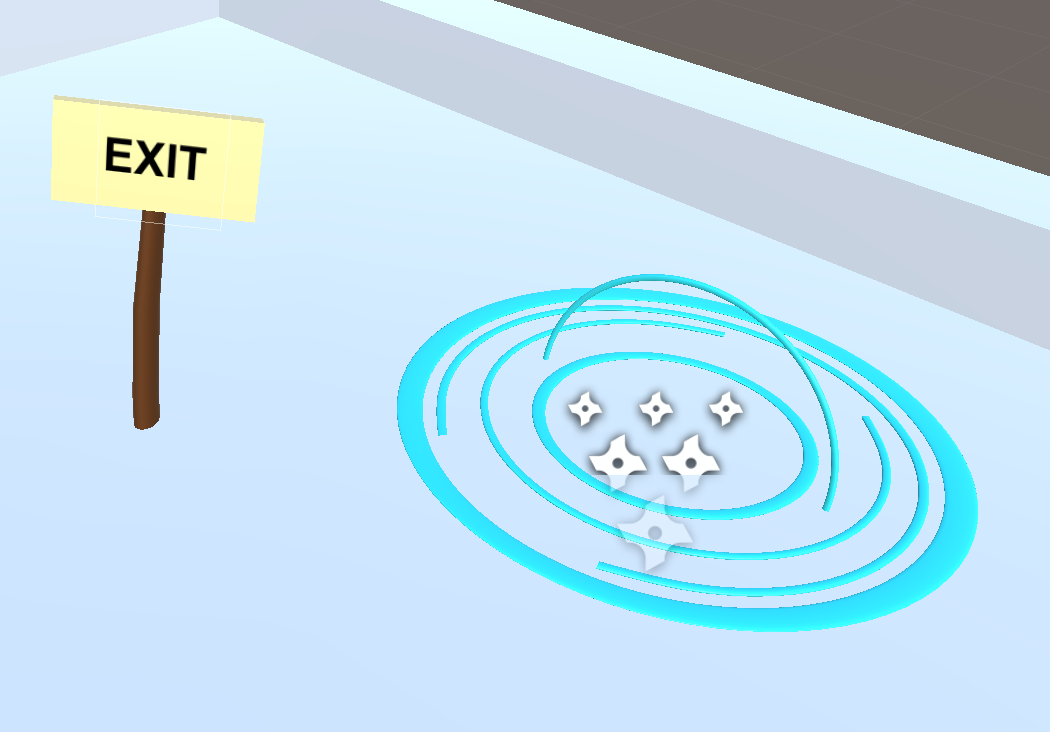
This explains why the long path in the Level 0-2 is a succession of small planes instead of a long unique one.



Design (Mid-Stage):

Going back to my core objectives, I wanted to have a simple game. Now that I have my mechanics ready (and intuitive as possible), I needed a way to force the player to use them. I wanted them to play without any tutorials.

To do so, I decided to Add instructions in the world telling them what they needed to do.

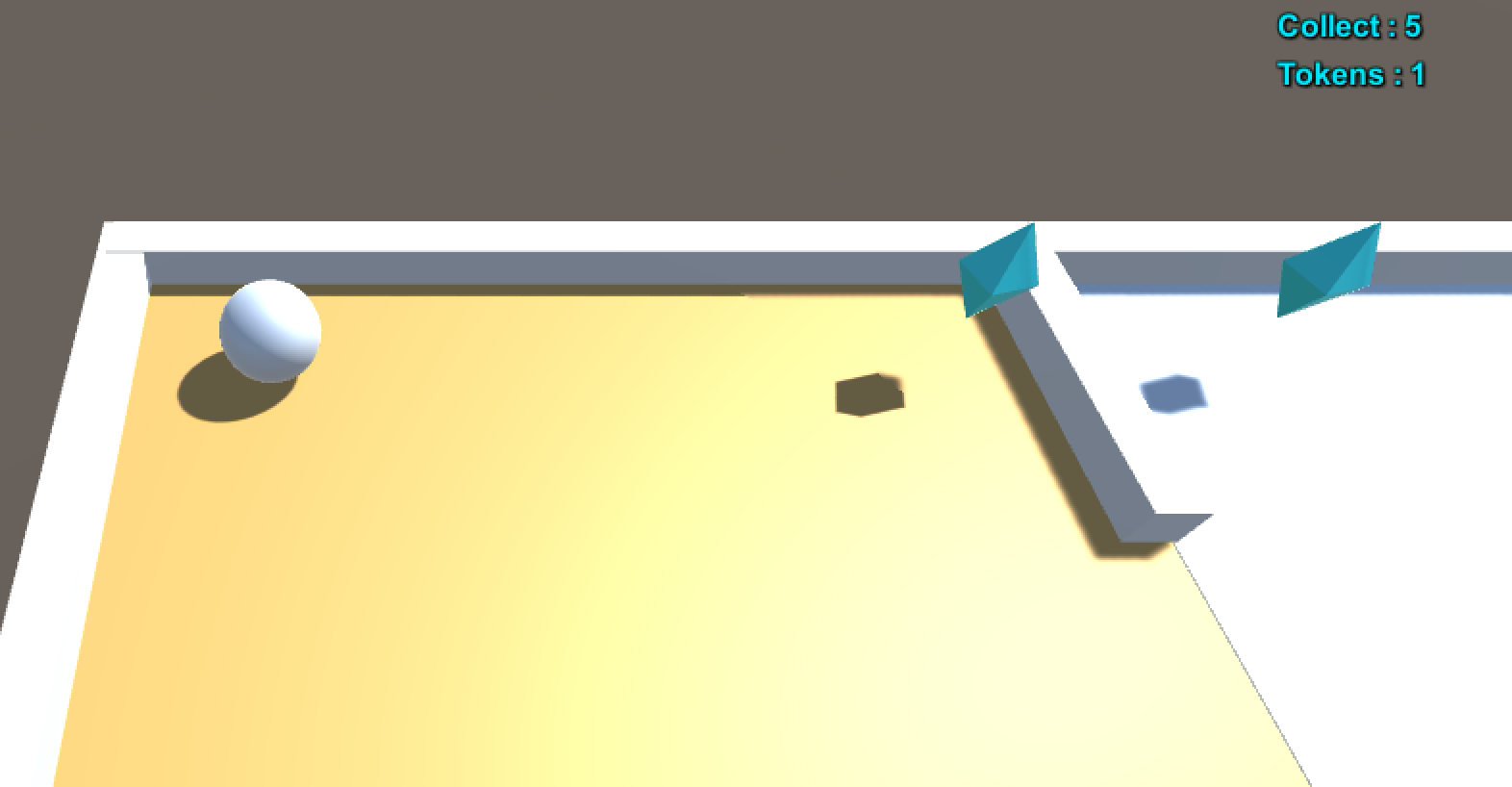
I created panels with “exit” or “jump” written.



I wanted to force the player to try things without teaching them the exact way to do it.

I made sure that the keys were the same as almost every game: “aswd” to move and “space” to jump. I hoped that these keys were intuitive enough to people.

I also added rotation and sound to tokens. I wanted to encourage the player to be curious and go toward them, then realise they needed to collect them with the text on top-left.



There is a different animation to the exit when it is ready to use and the colour of the ground indicates which platform you can or cannot move.

The first level was designed to teach the “collect token”, “exit” and moving to different plane mechanic (the core gameplay).

The second a “jump” and “falling off ground” tutorial.

The third one introduces the health mechanic and the attack.

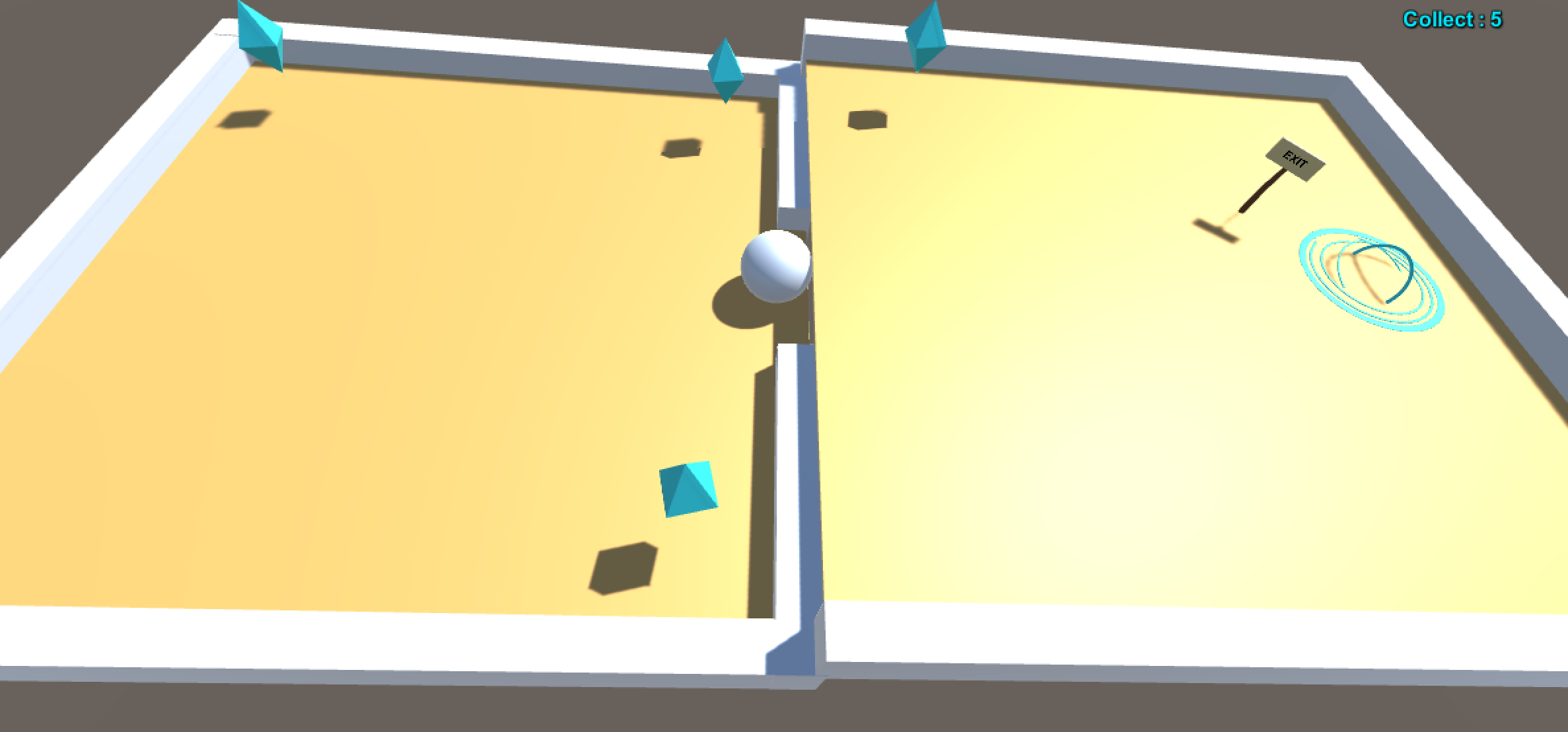
Design (Mid-Stage: Retrospection):

I thought everything was intuitive enough and made sense in term of design that anyone can understand the game without tutorial. And have fun with it.

I never was so wrong… I tested this game with many people, both gamers and non-gamers. None could understand how to play it without my help (and without frustration). The only noticeable difference was that gamers were learning faster than non-gamers.

The most common errors I saw were:

* Changing platforms: (~70% of about 15 people) had trouble understanding that the velocity of the ball carries even if the plane is not tilted. This is the first requirement to move plane and a lot of people stayed stuck in between like below.



* Jump mechanic: the jump (like the bouncing from walls) carry your actual velocity. It means that if we jump without velocity, the ball will stay idle. Lot of people (mostly non-gamers) had trouble understanding that they needed to build velocity before jumping.

Most non-gamers kept repeating the same action when it didn’t work. “It didn’t work last time but it might this time”. I witnessed people trying to force a jump without velocity over 10 times in a raw, not searching for a different way to solve the puzzle.

On the other side, gamers kept trying new techniques. “It didn’t work if I keep pressing this button, let’s try to jump this time”. They ultimately got the answer at the end.

I am sure that this is one of the main impacts of video games on people: They train us to explore every option available from the gameplay to advance through the level. Gamers adapts faster. If something doesn’t work once, or twice… it won’t work at all.

That, sadly, doesn’t apply to the gameplay’s learning process. Everyone needs to be thought the tools to play the game first before thinking of different situation to apply it. It will be hard (and frustrating) to find the good tools alone, then apply them.

Design (Late-Stage):

The feedback I got from the mid-stage test helped me optimising the game. I added a simple tutorial to each level and it became more accessible to people.

I then created a “Victory” and “Failure” condition to each level and it became of playable prototype. The one anyone can download now.

Project Final Analysis:

This project was supposed to be about Level Design (which explains the name “Level”). I wanted a Game Design as simple as possible so the Level will bring the difficulty, the unexpected and the fun.

I was aiming to teach mechanics slowly, that once I introduce a new one I am sure that the player already masters the previous one. The Player could solve puzzles differently using the skill set as they wished (jump above lasers, or roll when it’s safe to do so, etc…).

I learnt a lot about problem solving, and also understood I couldn’t solve everything perfectly. I had to compromise with the level’s design to make sure it fits the core design perfectly.

This project’s foundation is now complete. The puzzles can be created very easily, new enemies with different mechanics can be implemented and the game can now grow into a bigger, funnier version of itself. I might as well create a small story now… How about we go save a princess ;)?

Thank you for playing!

Cheers,

Alexandre