SpaceQbit

by Sparks Studio

Report n°1

Erwan Vivien
Christophe Terreaux
Zacharie Constans
Gautier Picard

Contents

1 Introduction

Since the beginning of this project we mainly focused on learning how to use Unity and the tools we prescribed for ourselves (Photon Bolt, Photoshop, etc.). This first part of our project's development was also centered on the base components of the game such as the movement of the character, the use of the weapon, the first graphics of the character, the online version of the game and the website layout.

These basics will allow us to implement the more ambitious parts of the project later such as the procedural generation of the maps.

1.1 Sparks Studio

1.2 SpaceQbit

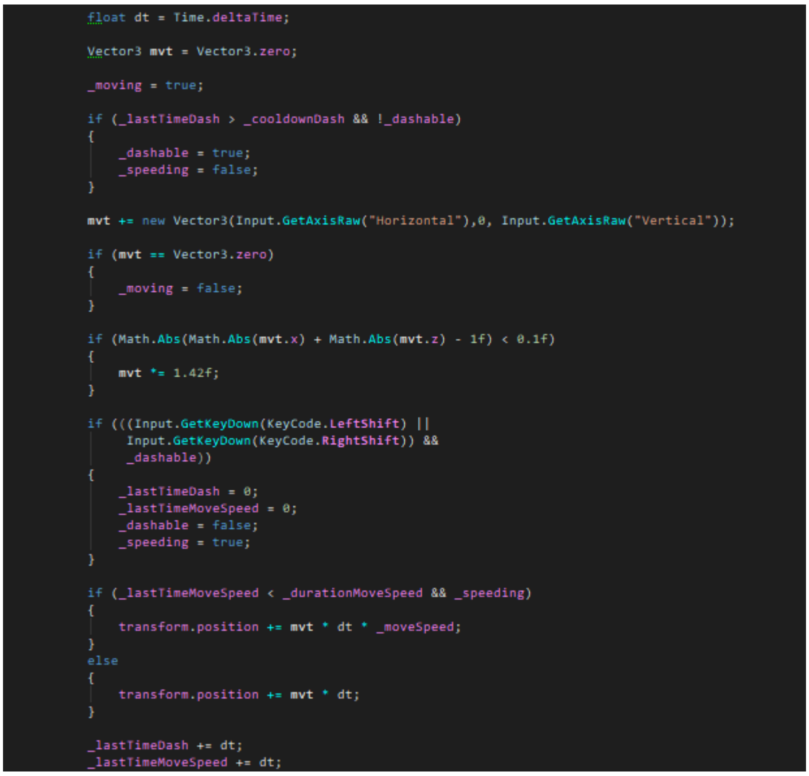
SpaceQbit is a game that takes place in... space. One is playing Bob, a mercenary who hunts creature through the universe. One day, during a mission (the tutorial), he finds a piece of an ancient alien technology. Not a big deal considering the fact that in space, everyone is an alien. However, this piece belongs to an old invention : the "SpaceQbit", a self-alimented powerful motor. This discovery is problematic, because this motor can either permits to escape everyone or upsetting the market of spatial ships' motor.

Bob will travel through many planets (approximately three) finding the different pieces in order to fix this technology.

During the adventure, many weapons will be available as well as upgrades for your character, thus allowing to have different play styles. When the game is finished, the player will unlock access to the "endless world", a procedural generated map, offering more and more challenge and rewards. The player can invite a friend whenever he wants. When two players are in the same game, the overall difficulty is raised, which means the enemies are stronger, but then the rewards are better.

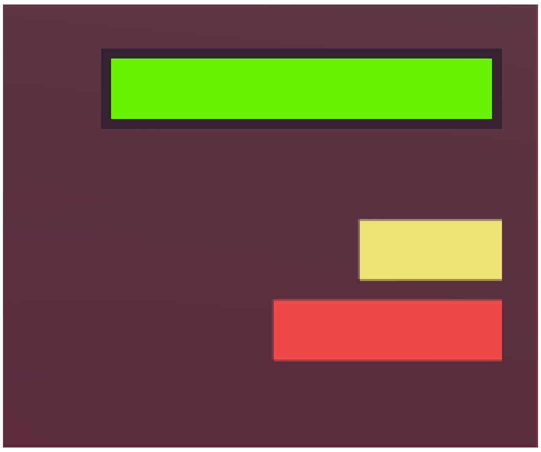
2 Progress

2.1 Script behaviours



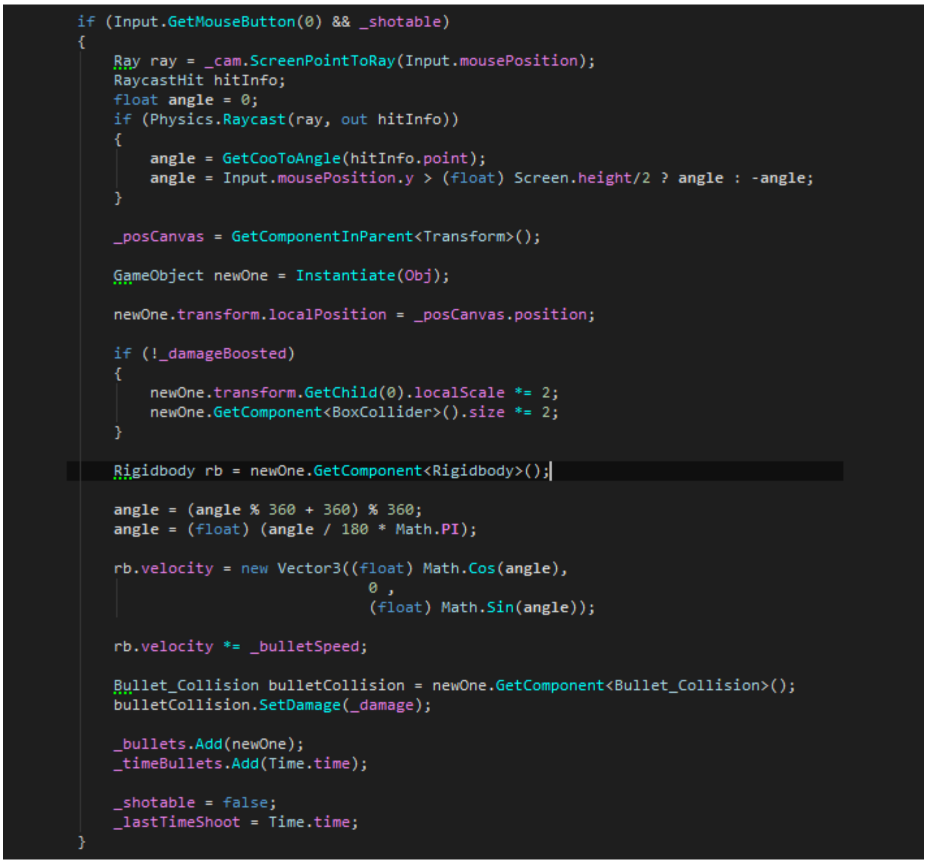
```
float dt = Time.deltaTime;

Vector3 mvt = Vector3.zero;

_moving = true;

if (_lastTimeDash > _cooldownDash && !_dashable)
{
    _dashable = true;
    _speeding = false;
}

mvt += new Vector3(Input.GetAxisRaw("Horizontal"),0, Input.GetAxisRaw("Vertical"));

if (mvt == Vector3.zero)
{
    _moving = false;
}

if (Math.Abs(Math.Abs(mvt.x) + Math.Abs(mvt.z) - 1f) < 0.1f)
{
    mvt *= 1.42f;
}

if (((Input.GetKeyDown(KeyCode.LeftShift) ||
     Input.GetKeyDown(KeyCode.RightShift)) &&
     _dashable))
{
    _lastTimeDash = 0;
    _lastTimeMoveSpeed = 0;
    _dashable = false;
    _speeding = true;
}

if (_lastTimeMoveSpeed < _durationMoveSpeed && _speeding)
{
    transform.position += mvt * dt * _moveSpeed;
}
else
{
    transform.position += mvt * dt;
}

_lastTimeDash += dt;
_lastTimeMoveSpeed += dt;
```

(a) Cooldown script



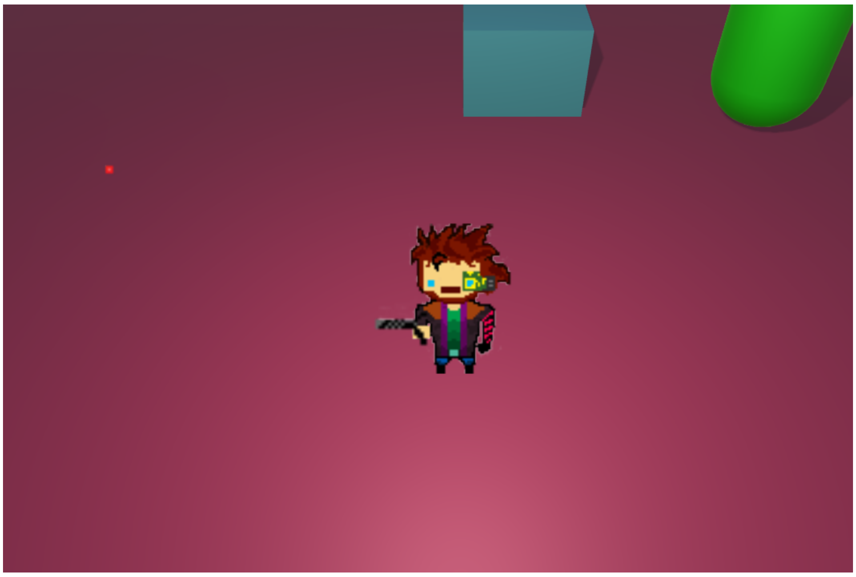
(b) Cooldown canvas

Here, the yellow bar is the sprint cooldown and the red one is the damage boost. When the player respectively presses the 'Shift' key and the 'R' key, he gets an acceleration momentum and a damage bonus. Since we don't want the player to 'spam' those keys, we added a visual timer.



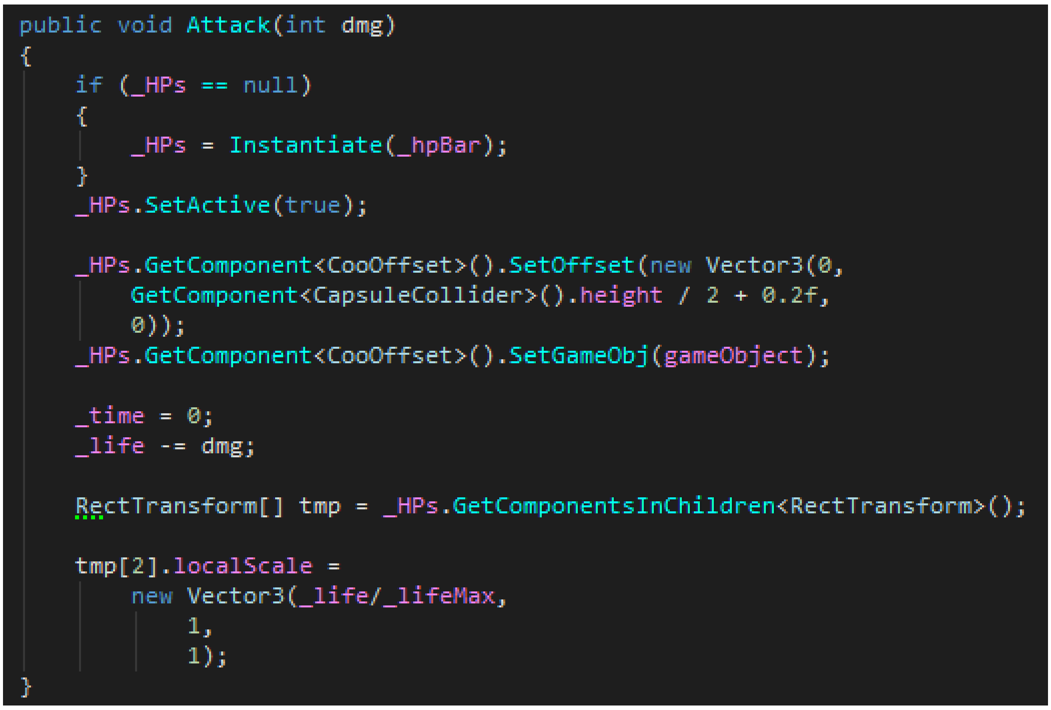
```
if (Input.GetMouseButton(0) && _shotable)
{
    Ray ray = _cam.ScreenPointToRay(Input.mousePosition);
    RaycastHit hitInfo;
    float angle = 0;
    if (Physics.Raycast(ray, out hitInfo))
    {
        angle = GetCooToAngle(hitInfo.point);
        angle = Input.mousePosition.y > (float) Screen.height/2 ? angle : -angle;
    }

    _posCanvas = GetComponentInParent<Transform>();

    GameObject newOne = Instantiate(Obj);

    newOne.transform.localPosition = _posCanvas.position;

    if (!_damageBoosted)
    {
        newOne.transform.GetChild(0).localScale *= 2;
        newOne.GetComponent<BoxCollider>().size *= 2;
    }

    Rigidbody rb = newOne.GetComponent<Rigidbody>();

    angle = (angle % 360 + 360) % 360;
    angle = (float) (angle / 180 * Math.PI);

    rb.velocity = new Vector3((float) Math.Cos(angle),
                                0 ,
                                (float) Math.Sin(angle));

    rb.velocity *= _bulletSpeed;

    Bullet_Collision bulletCollision = newOne.GetComponent<Bullet_Collision>();
    bulletCollision.SetDamage(_damage);

    _bullets.Add(newOne);
    _timeBullets.Add(Time.time);

    _shotable = false;
    _lastTimeShoot = Time.time;
}
```

(a) Script to shoot



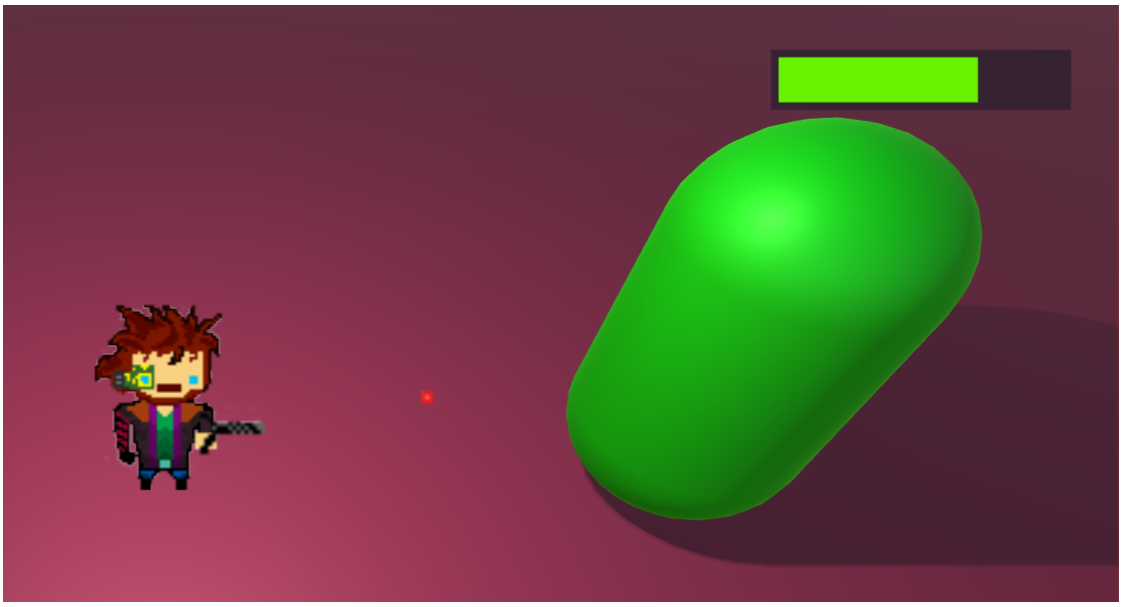
(b) Bullet example

We are having some issues to report the angles in different planes using mouse position. Since we are in a 45° plane and the mouse is in our 2D screen, we need to adapt the formulas.



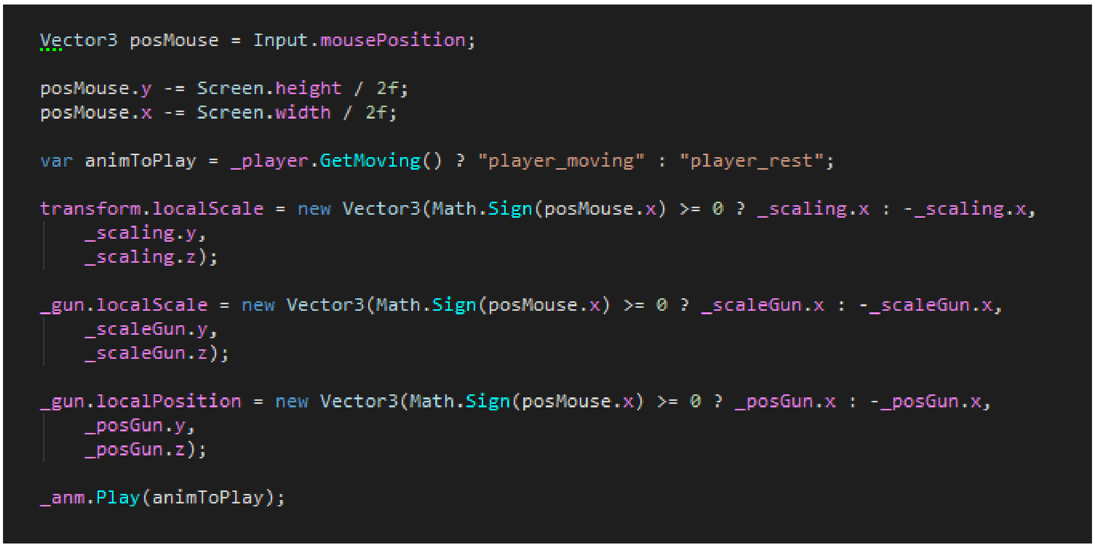
```
public void Attack(int dmg)
{
    if (_HPs == null)
    {
        _HPs = Instantiate(_hpBar);
    }
    _HPs.SetActive(true);

    _HPs.GetComponent<CooOffset>().SetOffset(new Vector3(0,
        GetComponent<CapsuleCollider>().height / 2 + 0.2f,
        0));
    _HPs.GetComponent<CooOffset>().SetGameObj(gameObject);

    _time = 0;
    _life -= dmg;

    RectTransform[] tmp = _HPs.GetComponentsInChildren<RectTransform>();

    tmp[2].localScale =
        new Vector3(_life/_lifeMax,
            1,
            1);
}
```

(a) Health bar script



(b) Health bar example

Here, we consider the capsule as the enemy. The red dot is the bullet and it's going to damage the enemy. After five seconds, the health bar disappear leaving the player more visibility.



```
Vector3 posMouse = Input.mousePosition;

posMouse.y -= Screen.height / 2f;
posMouse.x -= Screen.width / 2f;

var animToPlay = _player.GetMoving() ? "player_moving" : "player_rest";

transform.localScale = new Vector3(Math.Sign(posMouse.x) >= 0 ? _scaling.x : -_scaling.x,
    _scaling.y,
    _scaling.z);

_gun.localScale = new Vector3(Math.Sign(posMouse.x) >= 0 ? _scaleGun.x : -_scaleGun.x,
    _scaleGun.y,
    _scaleGun.z);

_gun.localPosition = new Vector3(Math.Sign(posMouse.x) >= 0 ? _posGun.x : -_posGun.x,
    _posGun.y,
    _posGun.z);

_anm.Play(animToPlay);
```

Figure 1: Looking direction script

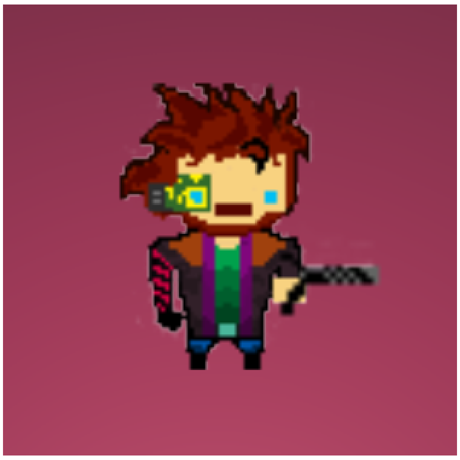


Figure 2: Right sided.

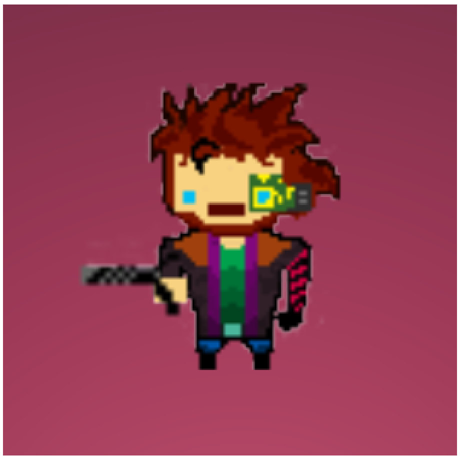


Figure 3: Left sided.

The script takes in input the mouse position relative to the screen and updates the scale vector always to look at the cursor. It is a simple feature we really wanted and here it is, working well.

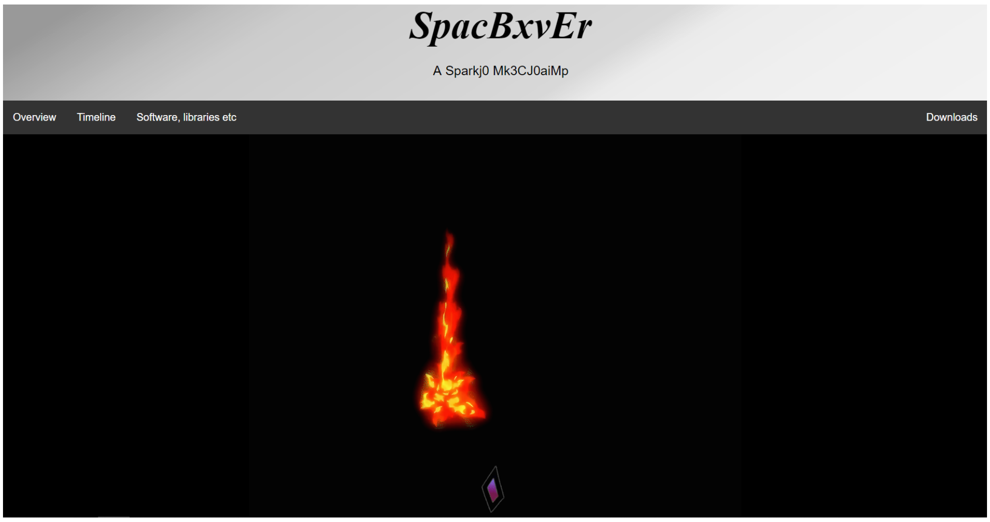
2.2 Website

The website was probably the simplest part of the project, as we expected. Thus, we really are ahead of the initial schedule from the book of specifications, without suffering from any delay concerning the other parts.

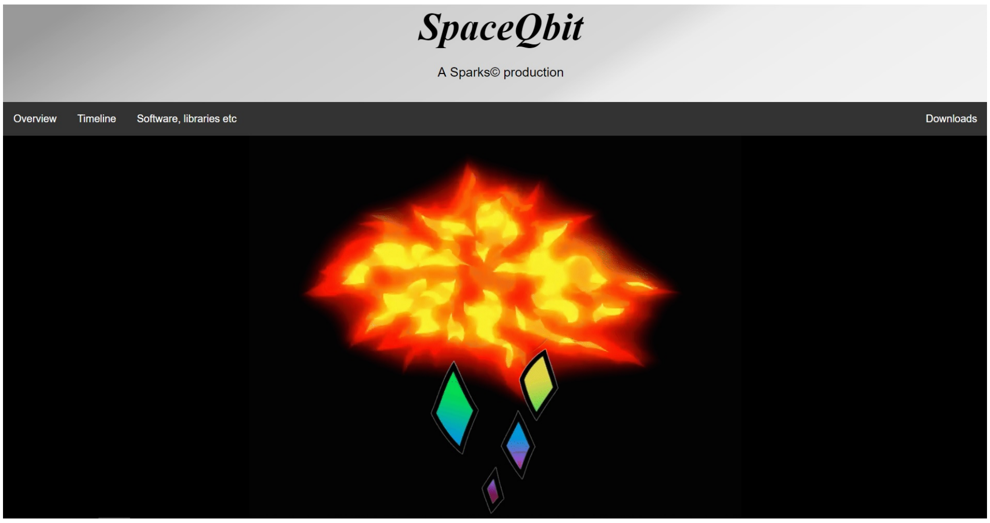
We completely made the website from scratch, using HTML, CSS and JavaScript. When we met to discuss about the website, we quickly agreed to make a quite simple one, but still good-looking that gives the player a taste of the game's atmosphere and universe. We waited to have a good idea of how the website should be done so we did not have to cancel everything and redo it.

Then, we started to work on the home page, which contains the animations (cf. animations below). The most complicated part was to structure the page and organise well all the elements it contains, but once it was done, we only had to copy this structure in each tab's code as they are all based on the same one. After, we had finished the home page, we added all the tabs at once (Overview, Timeline, Software and Libraries, Download) to have an overview of how the website would look when navigating through it.

The best was then to come. We could fill the website as we wanted and let course to our creativity concerning the design. We quickly thought about having an animation on the title also, only in the homepage. While navigating on internet, we found this effect of generating random letters until the word is complete, and we decided to implement it. Basically, the title is given two variables, update being the number of letters that will be generated before the right one, and duration, which, as the name says, is the time it takes for the right word to be generated, and the function will pick a random character for each letter of the world, and letter by letter, it will finally show the right word.

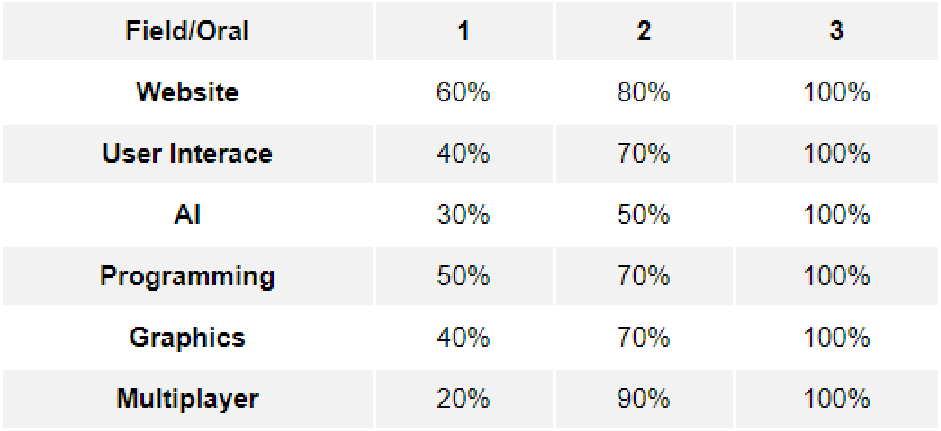


(a) Animations



(a) Animations

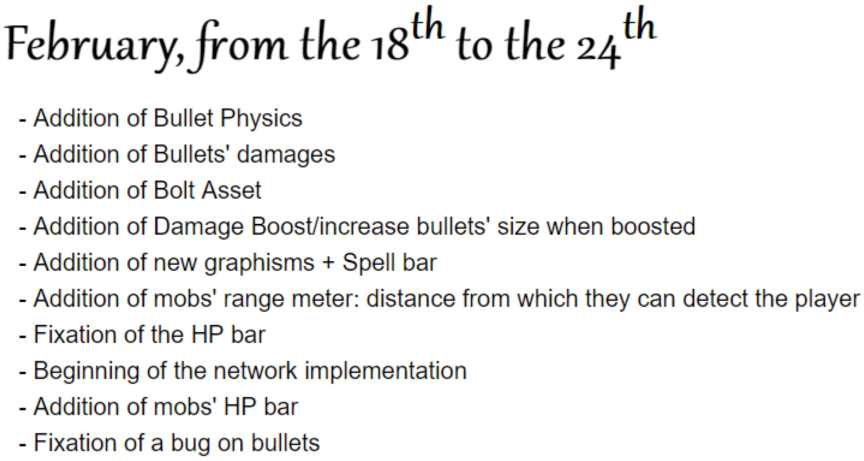
Consequently, we could not afford to simply "screenshot" the tables from the book of specifications and insert them in the tab overview. We chose a simple but efficient design (cf. tables below). It has been quite an issue to place them correctly on the page though. Then, for the members' presentation, we used our picture from EPITA, because they looked professional enough while still being casual, and we kept the same description as the ones in the book of specifications.



Field/Oral	1	2	3
Website	60%	80%	100%
User Interace	40%	70%	100%
AI	30%	50%	100%
Programming	50%	70%	100%
Graphics	40%	70%	100%
Multiplayer	20%	90%	100%

Table 2: 1st soutenance's update

Concerning the timeline, we knew it would have been painful to keep it until last minute, so we kept an history of every commit we made on git and filled the page progressively, each time we had add something consequent, sorted by week. That way, we did not have to retrieve two months of commit and to sort everything right before the presentation.



February, from the 18th to the 24th

- Addition of Bullet Physics
- Addition of Bullets' damages
- Addition of Bolt Asset
- Addition of Damage Boost/increase bullets' size when boosted
- Addition of new graphisms + Spell bar
- Addition of mobs' range meter: distance from which they can detect the player
- Fixation of the HP bar
- Beginning of the network implementation
- Addition of mobs' HP bar
- Fixation of a bug on bullets

The Software's tab is quite empty because we didn't have much to write inside. We are still thinking about how to make the tab look better, or maybe merge it with another one.

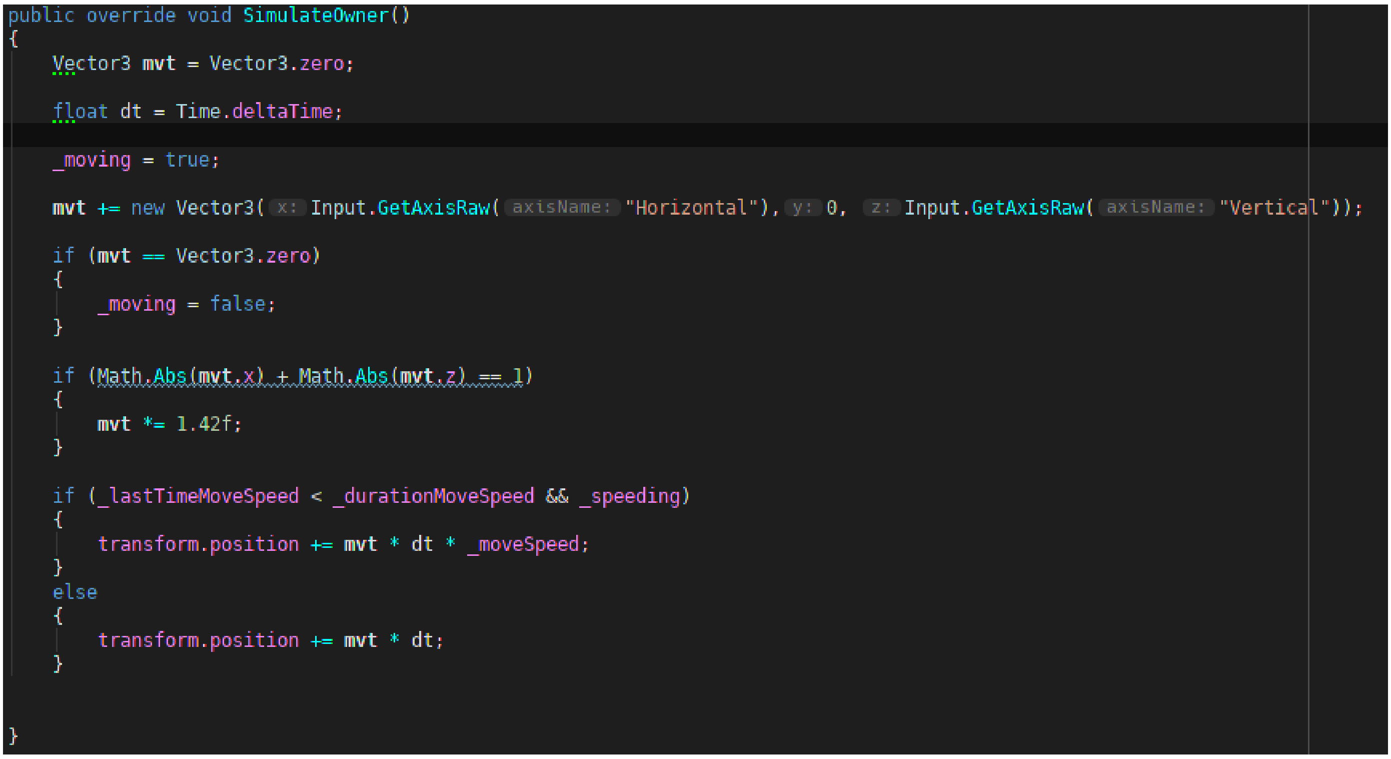
Finally, even if there is no use of the download tab yet, we decided to add a download button "for fun".

We are really happy with the result because it looks really clean to us, and is at least as good as what we imagined. Moreover, the website being almost done, it is one task less to think about and let us have more flexibility for the task distribution.

2.3 Network

The Network is constructed using the Photon Bolt Engine. Thus an important part of our work has been to learn how to use it and how to incorporate our scripts such that they behave the same way in online mode (with Bolt) as in offline mode (for which they were created in first place).

We started by following the official Bolt tutorial, but we encountered a few issues such as unreachable scenes that were not ’seen’ by rider despite being built by Unity’s compiler, a problem which repeated itself when we started the network implementation in the actual project.



```
public override void SimulateOwner()
{
    Vector3 mvt = Vector3.zero;

    float dt = Time.deltaTime;

    _moving = true;

    mvt += new Vector3( x: Input.GetAxisRaw( axisName: "Horizontal"), y: 0,  z: Input.GetAxisRaw( axisName: "Vertical"));

    if (mvt == Vector3.zero)
    {
        _moving = false;
    }

    if (Math.Abs(mvt.x) + Math.Abs(mvt.z) == 1)
    {
        mvt *= 1.42f;
    }

    if (_lastTimeMoveSpeed < _durationMoveSpeed && _speeding)
    {
        transform.position += mvt * dt * _moveSpeed;
    }
    else
    {
        transform.position += mvt * dt;
    }


}
```

Figure 4: The Simulate owner function regulating the movement of the player

Implementing bolt into the scripts is the actual stage of the networking. It mainly consists in changing the scripts so that the presence of a client does not change their behaviour.

A big part of this concerns modifying the player-related scripts using the function ’SimulateOwner’ which restrict the player able to move his character to the ’local’ player.

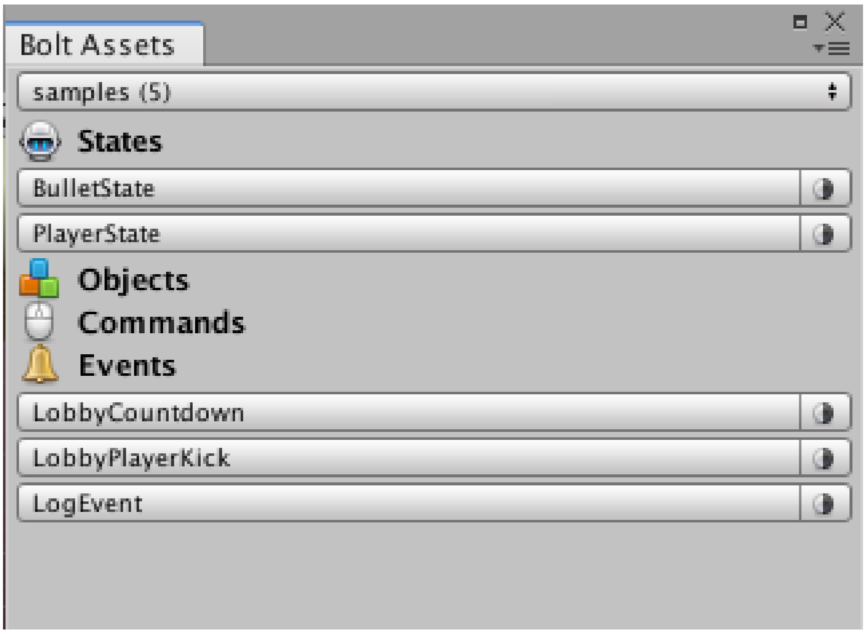


Figure 5: Bolt Entities

In bolt, game objects can be defined as ’States’ this definition is what allows them to replicate over the server, thus another part of the network implementation is the management of these states and their properties. So far the network allows the presence of different players on the same terrain but the each one of them does not have a working individual camera. And the movement animations of the players are yet to repeat themselves on the server.

2.4 Graphics

All the graphics were created with Adobe Photoshop. We chose similar proportion to the game "Enter the Gungeon" : our main character is composed of 49×62 pixels. The design of our main character went really fast. We had many inspirations, thus it is a mix-up between characters from different universe and games. With 2D models, we have simpler animations, which offer better readability and a nervous game.

We set-up the model after the one of the Pilot of "Enter the Gungeon", we shifted to a mix between the character "Big Boss" of the "Metal Gear" franchise and "Rhys" from "Tales from the Borderlands". Our final goal was to offer a character which is easy to recognise and not full of unnecessary details.

(a) Rhys

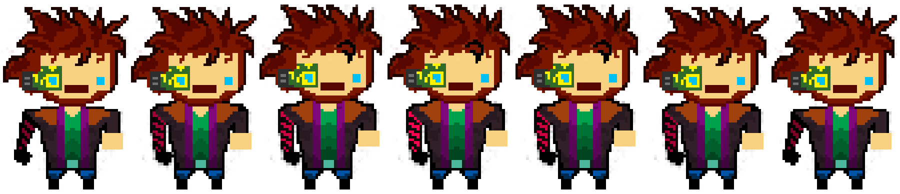
(b) Big Boss

(a) the Pilot

(a) Bob

We kept in mind the fact that our character has to wield weapons. Then, his left hand is motionless, but this part is subject to changes. The second step was the animations. Creating animations is something easy but also very frustrating. For example, creating a walking animation can be very easy but it will not be very stylish. That is why it is frustrating. The ease at the cost of the style.

Right after the walking animation came the standing animation. This one was easier because animating a breathing is simpler than giving a decent animation of walking. Creating an animation with Photoshop consist of creating many layers. Each layers correspond to a frame of the animations. When we consider that the animation is finished, we have to put all the layers on a big strip which size correspond to : (the size of a frame × the number of frames)



(a) Respiration sheet

We also began to think about the bestiary of the game. Our first enemy is a so-called "Mister Freeze". The bestiary is very important because it also decides what is the atmosphere of the planets. Thus, we chose to have an ice planet called "Frostise" and the second one called "Kowt-Kowt".

3 Realisation

3.1 Success

We kept a good atmosphere through the different months. Our biggest success is our website which is almost finished. Then Gautier, who was working on it, switched to another field of the project. The management of our resources added with the skills of the members led to a better progression of the project.

For now, our capsule is triggered in a 3 meter radius and follows us. In a 1 meter radius, it stops moving and it attacks the character every one second. Also it stops targeting the player if he goes 6 meter away from it.

The system of damage dealt works perfectly at the moment, where the bullets are dealing damage and there is a way to increase the damages using boost for the next two bullets.

We added the cooldown bars, and we only have to make them look fancy since it is only plain images for the moment.

3.2 Complication

We want to implement a shooting system but it is not a first view game and we are using the mouse coordinates to calculate what angles should the bullet take. However, the mouse position is in a different plane as the game plane and it is a struggle to change plane for angles. We still need to find a formula that fits perfectly.
For example:
-When he mouse gets close to the gun, we get some weird angles...
-When it is too far in the plane, the angle delta is at it's maximum...

We also had some problems using the Canvas system of unity. It was often disappearing while we were still able to see it in the scene. Many times we had to re-make the whole architecture of the canvas. Also, managing the in-game menu is a real pain. Every canvas must have his own parent linked to it. The escape key, which should be our go back key needs to handle every previous canvas etc... We have not managed to make it work perfectly for now.

The music is also a complication but a minor one. Considering the fact that we do not have any software to create quality sound and musics yet, it was impossible to progress concerning sound design.

Concerning the graphics the main complication was that it often looks like a total potato. Thus, We have to work over and over to polish the animation. It is not a big deal but it is lot of time that could have been use to other animation/character.

One big flow of the network is the variety of objects and attributes that are interconnected and thus need to be managed in unison with a number of other things. This means that every new implementation of bolt to an object or a scipt needs a few others.

4 Prevision

4.1 Script behaviours

We plan to correct the angle miscalculation because as said previously it is a bit hazardous. This is the worse part, there is too much mathematics involved and we cannot find a simple algorithm to help us through it.

We also want to add the "real" enemies, since we are only using a capsule at the moment. We already have some enemy model and we are thinking about how he could move itself (Bounce ? Simple lines ?)

We need to implement a functional in-game menu as a main game menu, where the player presses escape and he will have access to all of the settings (key bindings & volume & etc)

We really want to add a procedural generation of a map, maybe for now a really simple one. It would mainly be walls, some obstacles and random enemy placement.

Also our own health points are not decreasing even if our health is changed. We want to fix this as soon as possible to be able to make test for the death and respawn.

4.2 Website

The website in term of pages is ’done’, we only need to add CSS to make it more fancy. It should look more beautiful and attractive to the eye.

We need to link the .exe of the final game to the download button and a button to donwload the reports.

We also want to find a way to make the GIF look better.

Finally, we plan to make the website responsive so it overall looks better and does not suffer from switching from a computer to a different screen size, or even a smartphone

4.3 Network

The future network will allow the player to create a server, and to name it, allowing others players to join him.

A 'quit multiplayer' button will be added allowing a client to leave a server.

An offline mode will allow a player to play without having an internet connection.

The terrain will become replicable on the whole server allowing all the players to have the same map despite the generation being procedural.

The animation of the characters movement will be seen by every players.

4.4 Graphics

Concerning the animations, we plan to have more enemies, with their animation, maybe a small rework Bob's and a map for the next presentation. Also, more weapons with animations. We plan to change the weapon system but theses ideas are still work in progress.

The rework of Bob's animations is tied to the rework of the weapon system. We plan to make weapons wield by small drone types. This offer easier moving animation, thus permitting better visual. Also, with weapons levitating around the player, it will be easier to animate and look better, offering more designs choices.

5 Conclusion

Until now, all our deadlines are respected. The members of the group are still very bound and there is no animosity. It is very important to keep this good ambience and this hard-work. We did face complication but it was not overwhelming. Globally, we are proud of what we have build so far.

Our project possess strong root and this is what we needed for this first presentation. Having a strong skeleton for the following implementation will be, for sure, an advantage. The beginning was a little bit thought but know, we all know how to use our new tools such as Unity, Bolt, Photoshop and Rider. Thus, the next implementation will be easier.

Furthermore, we are all relieved that we did not have to rush our project. Working seriously since we were allowed was the right strategy. With that in mind, we will take a short break from the project in order to rework on it, full of energy and determination.

6 Index