Interactive Graphics Project

Ionut Marian Motoi (1892355), Veronica Romano (1580844),
Leonardo Saraceni (1920088)

July 26, 2020

Contents

Chapter 1

The Concept

The inspiration for this project, comes from the classical platform games, like for example Super Mario and Kirby, but in our case the hero would simply be a lonely knight. To finish each level, the task to accomplish is to collect all the three coins spread throughout the level, to activate the portal and reach the next level. Obviously, the difficulty is given by the fact that there are only three lives available, and if all the lives are lost, the game is over. There are two possible ways to lose lives. The first is to fall off a platform, and the second one is to go on a trap. To make the things easier, since the levels are quite long, there are some intermediate checkpoints, from the which the player restart the game once he falls off a platform. The target was to create an "old style" platform game with suggestive environments and effects, therefore we created four levels (one is the tutorial), each of which corresponds to a season of the year (spring, winter and so on), with different effects, models and also game dynamics, offering a wide range of unique peculiarities. The difficulty increases with each level with harder paths to go through in order to find and reach the coins. The coins can be hard to find, since they are not visible at first sight, therefore it is necessary to explore all the corners of each level to find them.

Chapter 2

Framework

Initially, the framework used to create the game, was **Threejs**, so we began to explore it, but we found lots of difficulties in the definition of the collisions between the player and the models. In order to detect the collisions, we had to build up a **ray caster** mechanism, that wasn't working really well in **Threejs**, since was really hard to manage different rays. The other option was to use the **Physijs** plugin for **Threejs**, but it seemed an overkill work, since the task we had to accomplish were quite easy, and from a computational point of view it would add too much work. In fact there was no need for realistic destructions of the objects, or fluid motions. Therefore after a few days, and after having read lots of documentation, we decided to completely change it, and to explore **Babylonjs**. This other framework, despite being quite a new library with respect to **Threejs**, is what we really were looking for. It is intuitive (as **Threejs**), provides lots of useful tools like for example the **sandbox** and the **playground** where lots of examples are defined and is possible to learn from them and modify in real time the code in simple scenes and then apply what we learned to our project, and it allows to explore many different functions. Moreover, there is a really active community behind **Babylonjs**, and this is good since is easy to find the solution to some problems on different forums. This framework, is really flexible and therefore for these reasons it has been our choice for developing this project.

Chapter 3

Controls

As in many games, is possible to move horizontally using the keys "A" and "D", respectively to the left and to the right, while to jump we need to press "space bar", and the character will jump, obviously, only if it is on a platform. To give the gameplay a bit of variety, pushing the button "P" is possible to activate the "run mode" through the which the player can make some particular jumps, that otherwise would be impossible to do just walking. The idea was to combine the controls of every classic platform game, with the very basic equations of physics. In this way, is possible to achieve an effect that gives both reactive controls, and the acceleration effects. It was quite hard to implement this, but once it was done, it was possible to simply modify the acceleration (and the deceleration) given to the player, depending on the kind of platform, and more specifically based on its material. For example, if the platform is made of ice, the player tends to slide on it, making the gameplay way more challenging, especially on little platforms. Therefore is strongly recommended to don't run on them. Moreover, pressing the "P" button of the keyboard, is possible to make the player run, increasing the acceleration of the player, respect to when he only walks. This is fundamental because in order to do some particular jumps for the which is required a higher velocity. Another feature, is that , for each material an acceleration has been defined, going to simulate the friction effect of different surfaces. In such a way, the velocity and therefore the position of the player changes, following the kinematic Newton law $X = 0.5AT^2 + V_0T + X_0$.

It was hard to build a system of timers, that could manage both the velocities along the X axis and the Y one, that as i said before, are important since are necessary to implement the Newton law, making the player position time-variant. The full control system is based on a **ray-caster** mechanism, that consists of a total of 6 rays, used to check all the collisions and the contacts of the player. One ray checks for the upper collisions, in order to avoid the character to go through platforms during jumps, and other three are used to check the lower collisions with the ground, so that when the player is touching the platform, it is able to jump. If the player is not touching the ground, and is in mid-air, the player is clearly not able to make another jump. Three rays are used and not a single one, disposed at the middle and at the right and left edge of the body, in order to have an higher accuracy in the jumps, because otherwise at the edge it is not possible to jump. Here is a picture that shows how the rays are distributed. Is important to notice that this system, allow us

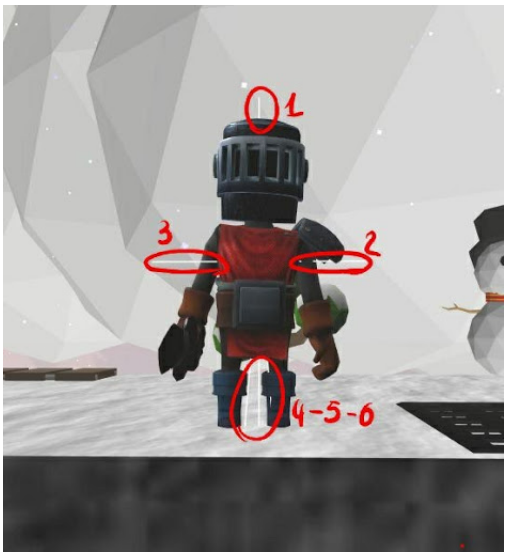


Figure 3.1:

to pick the property and the **id** of the material encountered, allowing to manage the accelerations and the velocities of the player depending on the surface he is. Moreover, it is important also to trigger lots of events, like the traps, the firebones and the portal. These are the essential steps of the control system, on which the game is based on.

Chapter 4

Camera

The camera that was used for this game is a Follow Camera, a type of camera implemented in Babylonjs that takes a mesh as a target and follows it as it moves. This type of camera is perfect for a platform game since we need only to assign the knight as a target and then set the parameters of the camera until we are satisfied with the result. Some of the available parameters of the camera is the mesh to follow, the distance from the mesh, a height offset and a rotation offset.

Chapter 5

Level Design

What we thought was to recreate what could be the life of a knight. That's why we've set up a game based on four levels, one for each season of the year. In this way our knight can face different environments and climates, but also different types of obstacles such as rain, snow or ice. Our year starts in the summer with the first level. This is a simple tutorial with which the player can learn the commands of the game and how to deal with the unexpected during the course as traps or changes of terrain. After the summer will come the first real level of the game, set in autumn, where the player will have to test himself and face adversity. After autumn will come the winter level, and finally the spring level is the hardest one. It is important to notice how every level is characterized by different features, like for example difficult jumps, different materials to deal with and so on. These will be analyzed later more in detail, in the description of each level. For simplicity all the levels consist of only three types of platforms, a small, a medium and a large one. In each level then only two types of texture was used. This allowed us to correctly map the textures on those three different sizes. In each level are placed models of objects in gltf format that are characteristic for each level as we will see in the following sections where we will explain which models have been used in each of them. There are also models common to all levels. These are the coins that are three for each of them, the checkpoint bonfires that are two and a portal that will be activated when all three coins are collected. In the following sections we will see the specific details of each level.

5.1 Particles System

A feature provided by the Babylon JS framework, is the particle system, that gives the possibility to spread a certain number of particles from an emitter. The interesting thing is that the emitter can be either a fixed point, or a moving one, the important is that this is specified by a 3 components vector, that represents a point in the space. So it can be attached to a mesh or even to the camera. Is possible to specify a maximum number of particles that are present at each time, in this way is possible to avoid a too big payload on the GPU or the CPU, but the amount emitted at each frame, is specified by the emitting rate. There are also two directions to specify, this means that the particles are spread

in a space between those two directions. Then, there are lots of features that is possible to assign to the particles, in particular the texture to assign, the maximum and the minimum lifetime, the linear and angular velocity, the color assigned at the beginning, the mid and the end of its life, and also the size of the window in which the particles are diffused. This system has been used and adapted multiple times in this project, since it allows to create great effects, like for example the rain, the snow, the fire, the coin blinks, and also the portal effect.

5.2 Skybox

For the background, in each level (except the autumn level) a skybox has been defined, each one of them created using six different images. The skybox is a graphical element, that consists of creating a really big cube, with an image (texture) attached on the each of the inside faces of the cube. Therefore, six images are used for the cube, inside of which the whole environment of the level is placed. It is clear that, all the six images placed on the cube faces, have to be correlated, in the sense that there has to be continuity between adjacent faces. In this way is possible to give the idea that the player is perfectly surrounded by a true environment.

Figure 5.1: Model of a skybox.

The images that form the skybox, form a **cubemap**, that is built from a single image, taking square pictures of the same size. In this way, is possible to compose the cube, and here is shown an example of a cubemap. Then for each level a different image has been constructed using online tools that create the correct file starting from our chosen images, in order to give the right setting.

5.3 Turorial

This is the first level that the player will meet, is the easiest one, since is just an opening to the fantastic world we created. The aim of this introduction is to make the player confident with the platforms, the jumps and all the game mechanics that will be encountered during the various levels. Tips and explanation of the various parts pop up as windows, where inside there is written all the

necessary information. In order to close those windows, is sufficient to just click on them with the mouse. The messages are activated by some platforms, that are not visible, and that don't collide with the player. A particular material (called **tutorialM**) has been associated to the platforms, that is picked by the ray-caster system, that triggers the function corresponding to the correct tutorial, by picking also the **id** of the object. In this way 6 different messages have been created. The idea, is that in this way the player can not only passively read some instructions, but also try what he reads, making the gameplay more interactive and efficient, since it can try multiple times the different parts until he succeeds. More in details, the 6 parts of the instructions are divided in the following way:

1. Introduction and first movements
2. Checkpoints (bonfires)
3. Traps and lives
4. Sand and other materials
5. Long jumps and run
6. Coins and portal

Going more in details, the environment of this level is the summer, represented by moving waves on the background, and a sun model that moves along the **x** direction with the player. As said before, the background is given by a skybox, with attached a cubemap of the waves image, that moves continuously, rotating around the **y axis**, of a predetermined quantity, in order to give more dynamicity to the level. Different models related to the summer have been placed all along the level, and bright colors have been used, with lots of illumination. The sun is a particular element since is characterized by an emissive texture, that helps the natural lights of the level to shine the environment.

5.4 Level1

For the first level we tried to reproduce an autumnal environment. For this purpose we searched for models and textures that remind of this season, such as trees with yellow leaves, typical of that period, but also other models that can be found all year round, for instance rocks, evergreen trees, fences and wells. Two textures were used for the platforms, both of them without movement modifiers to keep the first level easy enough. For the textures also a normal map was used to give a more tridimensional appearance to the platforms. To create a more suggestive environment less intense hemispheric light was used so that to reproduce a night setting while maintaining visibility. Clouds were added by using a texture applied to 10 planes with transparency that were randomly added on the background of the level and that slowly move during the level. Rain was added initially using a built-in particle-helper, but the resulting effect wasnt realistic nor efficient. So instead the texture of a raindrop was used in the particle system of babylonjs. The emitter of the particles was attached to the camera in such a way that it moves with the player, while being

Figure 5.2: Models used in the tutorial level.

efficient since the particles dont have to be generated in all the level but only around the player. In fact the particle emitter that was used was a box large enough to be bigger than the FOV so that even when running there are still raindrops on the screen without empty spaces. Many parameters such as scale, lifetime, rotation, emission rate and many more were set by trial and error until a satisfactory effect was reached. Also a rain background sound is played throughout the level. Lightnings were added to the level in a stochastic way by generating a pointlight with a random intensity (between certain values) in a random position among the clouds. The pointlight was kept in place for 5 frames so to be visible enough. The normal map in this case contributes to a more stunning effect overall. Finally a thunder sound was played choosing between 3 sounds based on the intensity of the lightning (with a more intense light a more powerful sound was used).

5.5 Level2

The environment of the second level, is the winter, therefore two main kind of materials have been defined: the snow and the ice. The snow is considered as a normal ground, therefore the character acceleration and velocity arent affected by it, while on the other hand, the ice is really tricky since on it the player has a really low acceleration and tends to slide when it changes direction. moreover it ever has to stabilize itself, otherwise it keeps going on, until it falls on. This becomes really hard to deal with, especially on the smallest platforms, but after a few attempts is easy to understand how to deal with them. To build the ambientation, lots of invernal models, like snowy trees, polar bears, penguins and snowman have been placed along the level, and also a playful animation has been done for the snowman, that keeps moving his arm. To give an awesome

Figure 5.3: Models used in the first level.

effect, a particle system is active on this level, that simulates the snow.

This system has a limited number of particles that are generated each second, and moves with the camera, so that at each moment is visible. Moreover, there are cloud models on the background that keeps moving along right, and regenerate when they reach the end of the world. The last environment characterizing element, is the fog, that gives a bit the effect of the high altitude mountain, that blurs a bit also the skybox background. This level is characterized by hard jumps and requires the ability to keep the control of the player, also on small ice platforms.

5.6 Level3

As mentioned above the third level represents spring and you can immediately notice the brighter colors and a much less impervious atmosphere than the previous levels. Going into technical details, we started the realization of the level starting from the structure of the platforms. The platforms are textured like a lawn by creating a multi material so that the effect of the texture on the objects is more three-dimensional rather than flat. The layer is quite dynamic because the platforms are placed at different heights and alternative paths have been created to reach the same point. An additional element to the other layers,

Figure 5.4: Models used in the second level.

at least as far as the basic structure is concerned, are the rotating platforms. These are created in the same way as the others but a continuous rotation around the y-axis has been added. Also to add an additional difficulty if the player is not in the center of the platform he will fall down and lose a life. To create the spring setting, the level has been filled with green vegetation patterns and colorful animals such as butterflies, birds and deer. The patterns were found online, and colored through blender software. They were then exported as gltf models and placed in the level. A special feature is towards the end of the level where giant apples can be found. These are created like the other models but a transparent box of material has also been created to make them walkable. In this way they are part of the knight's path itself. The characteristic models of the level are those shown below.

There are also other models that are common to all levels, such as coin, traps, bonfires and portals.

Figure 5.5: Models used in the third level.

Chapter 6

Animations

The animations were developed using the Animation object of **Babylon Js** that is based on a collection of keys specifying the parameters of a mesh at different frames for the duration of the animation. The main focus of this part was on the playable character since it needed an animation for walking in particular, but also to smoothly rotate from the current orientation to the orientation imposed by the the user through A/D. Also a breathing animation was added that slightly moves the trunk of the knight to make it more alive. Then we also added a rotation loop animation to the coins and a disappearance one for when collecting a coin. The triggering of the traps were also done through an animation. Finally to the snowmen in the winter level was given an animation that makes them wave continuously.

Chapter 7

HUD

The HUD has been developed using the GUI module of Babylonjs. It is 1 During the game, the user is prompted with 2 types of information at any time. The number of lives remaining in the upper right corner and the number of coins collected in the upper left corner. The presence or absence of these two types of features is expressed through the transparency of the images representing the coins and the hearts. The HUD has been also exploited in order to make appear the messages in the tutorial, that is discussed above.

Chapter 8

Music

Regarding the musics and sounds, we put a different background soundtrack on each level. For each of this the sound chosen is engaging and in theme with the season that the level represents. Also the musics give a sense of adventure and difficulties that the knight has to face up to, like a blizzard (winter season) or a storm with thunder and lightning (autumn). Other sounds are superimposed on the background music, like the sound of the thunder and rain in autumn level or the tweet of the birds in spring level. All sounds are balanced in such a way that one does not impose itself on the others. Other sounds inthe levels are linked to action that the knight is performing. For instance a sound is played when the knight jumps or when it takes the coin. This sounds are linked to the action that the knight performs. One last sound is that of the ending portal. This sound is linked to the mesh of the portal, in this way the sounds becomes more evident when the player is close to it and so to the end of the level. This choice is to guide the player to the end because some portals are more hidden than others. Finally the menu has a simple and classical soundtrack in the style of similar platform games.

Chapter 9

Menu

What game would it be without a menu?! The menu is the first part of the game with which the player interfaces. For this reason we have made it clear and simple and also pleasing to the eye. The main menu is very simple and contains what is essential. The first start button with which you can start playing, will open the first level, the summer one, consisting of a simple and fast, but effective tutorial. After the tutorial you can then access the first real level of the game. The second button of the main menu will open a second menu from which you can choose the level you want to play. Here there are 5 other buttons, four of them are the name of the seasons and clicking on them will open the corresponding level. The fifth button is a back button that allows you to go back to the main menu without having to use the web page control. The last button we find on the main menu is the credits button. From here you will simply open another page of the credits menu where you can find references to the course and the project itself, the names of the creators of the game, i.e. the three of us, and the academic year. Also here there is a back button to return to the main menu. The background images chosen are very suggestive and also these, like the musics, represent adventurous settings in which we can imagine that our knight will find himself during his intrepid life. The adventurous theme is also suggested by the name of the game placed in the background of the main menu: "The Knight's Adventures". The inscription was created by us especially for this game. The choice of colors such as pink or purple might suggest that our knight is not brave and ready to face danger, but trust us, try to believe! These are the functions you can find in the game menu. To make it we created a css file where we defined all the graphic parameters for its creation, from colors to positions, some transition effects. In particular for these effects we have chosen a scale transition, realized through the hover action in the css file. When we pass over the buttons that in our case are made up of the writings themselves, they double their size and change color. There is not much more to say, the result can be seen and it is really not bad!

Chapter 10

User Manual

10.1 Main Menu

Figure 10.1: Screenshot of main menu.

"START" → Click start to enter the game. The first level will be a tutorial that will guide you through the basic mechanics of the game.

"LEVEL SELECT" → Will bring you to a menu where you can choose which level to play.

"CREDITS" → Will show some information about the developers.

10.2 Level Select

Click on one of the four levels to start that specific level. Click on "BACK" to return to the main menu.

Figure 10.2: Screenshot of level select menu.

10.3 Credits

Figure 10.3: Screenshot of credits.

10.4 Game Interface

This is the interface of one of the levels. You control the knight which will be always at the center of your screen. Use A to move to the left. Use D to move to the right. Use spacebar to jump. Use P + A or P + D to run to the left or to the right respectively.

Figure 10.4: Screenshot of framework in winter level.

In the upper left corner you can see how many coins you still have to collect (3 in the image). The collected coins are opaque while the missing coins are transparent.

In the upper right corner you can see how many lives you still have. When all the lives are lost you have to start the level again.

In the upper part of the screen there is a button ☰ that will open a menu:

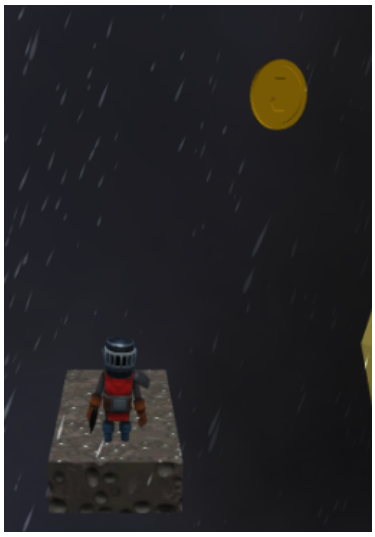
Figure 10.5: Menu.

Restart → will start the level again Main Menu → will bring you to the main menu Level Select → will bring you to the level select menu Close → will close this menu and you can continue to play

10.5 Gameplay

Collect all the 3 coins that you can find throughout the level, avoid traps and dont fall off the platforms. After collecting all the coins go to the portal to finish the level.

Collect coins by walking over them



Fireplaces are respawn points where you will respawn in case you fall off a platform

Avoid the traps or you will lose one life

Also falling off the platforms will cause you to lose one life

After falling off a platform you will respawn to the beginning of the level or to the last fireplace that you reached.

If you lose all your lives its game over and you have to start the level again.

After you collected all the coins the portal activates. Follow the sound to find it and reach it to finish the level