# Semestre5_FinaleProject_FlappyBird

Flappy Bird final project C++

## Overview

This is the C++ final project, and in this project we (IBOURK Youssef & EZ-ZAHER Manal) were asked to choose a mini-game that we will code with Cpp. The choice of the game was **Flappy Bird**, so we will try to make an approximate clone of this game. The real **Flappy Bird** game, was released in May of 2013 but received a sudden spike in popularity in early 2014 and became a sleeper hit At the end of January 2014, it was the most downloaded free game in the App Store **Flappy Bird** was removed from both the App Store and Google Play by its creator on February 10, 2014. He claims that he felt guilt over what he considered to be its addictive nature and overuse. The game was a real fun, and since its disappearing, we hardly wanted to play it, so we can say that, this is among the reasons why we have chosen to create this game, even if there many duplicates of it the internet.

To make this project stand we used the help of our ressources from the courses we took in class and we used also the help of "Qt Documentation".

## Concept of the game

The concept sounded too simple: Tap the screen to fly up, release to dive down, and maneuver through gaps in a series of green pipes clearly styled after those in the Super Mario series. The gaps were invitingly wide, many times the height of the bird. But because the bird moved so fast and dove up and down so quickly, making it through the gap without wiping out proved extremely challenging. Because you get just one point for each pipe cleared, your high score is likely to be in single digits, if not zero.

*Flappy Bird* To try the game online.

## PLAN

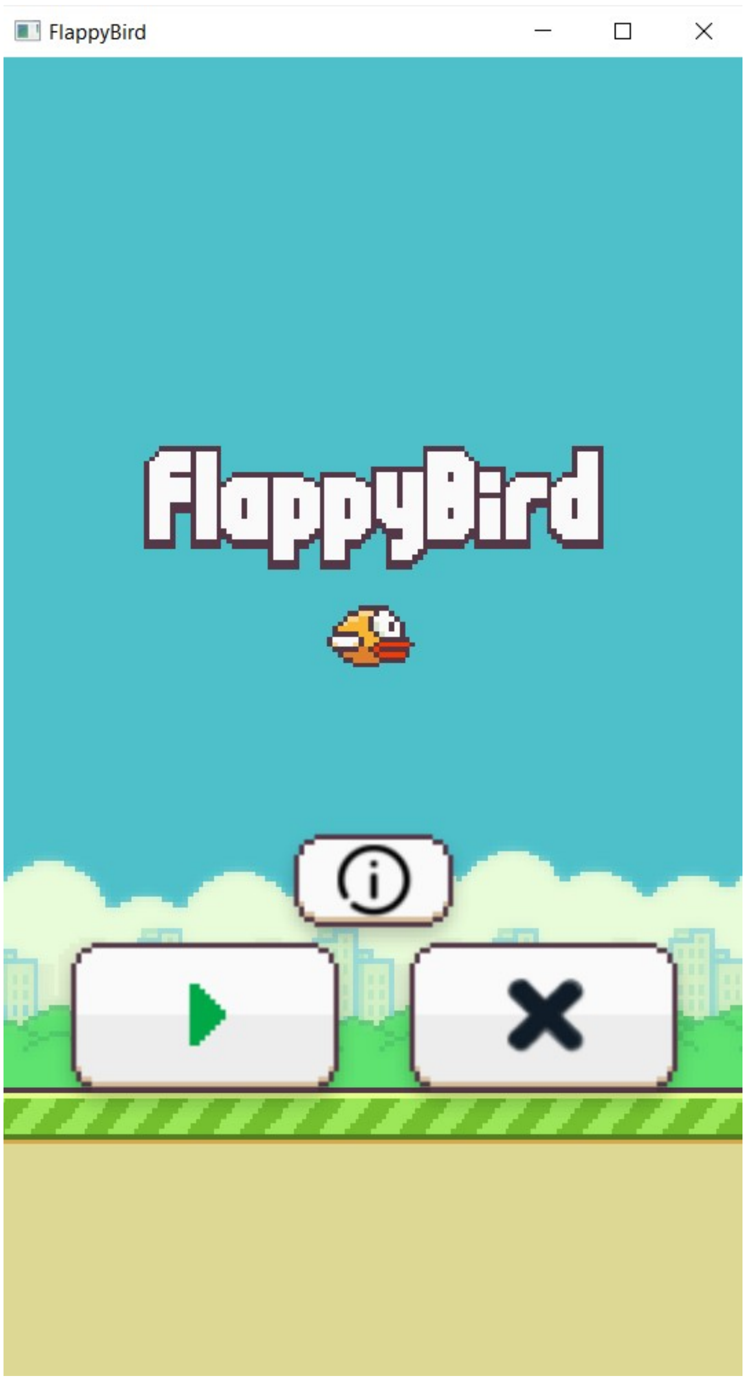
In order to start coding, and basing on our previous experiences, the process of programming should go according to a well chained plan

1. Collection of prequisites;
2. Ressources;
3. Coding;
4. Conclusion.

### 1. Collection of prequisites:

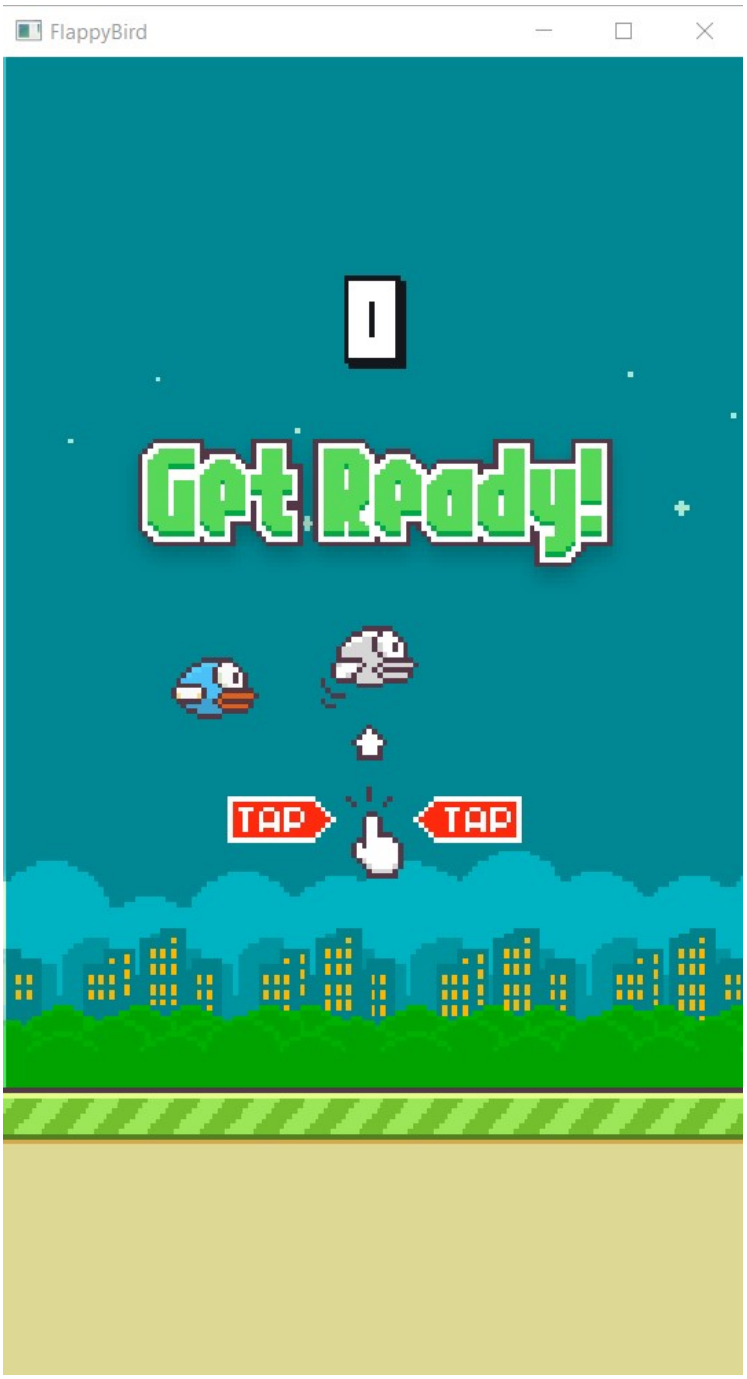
In order to start coding the game, we need to gather some prerequisites, the game will be devided into principal windows or interfaces, we will call them next by (GameStatus).

- "First Interface (GAMETITLE)": once we launch the game, this interface should appear,



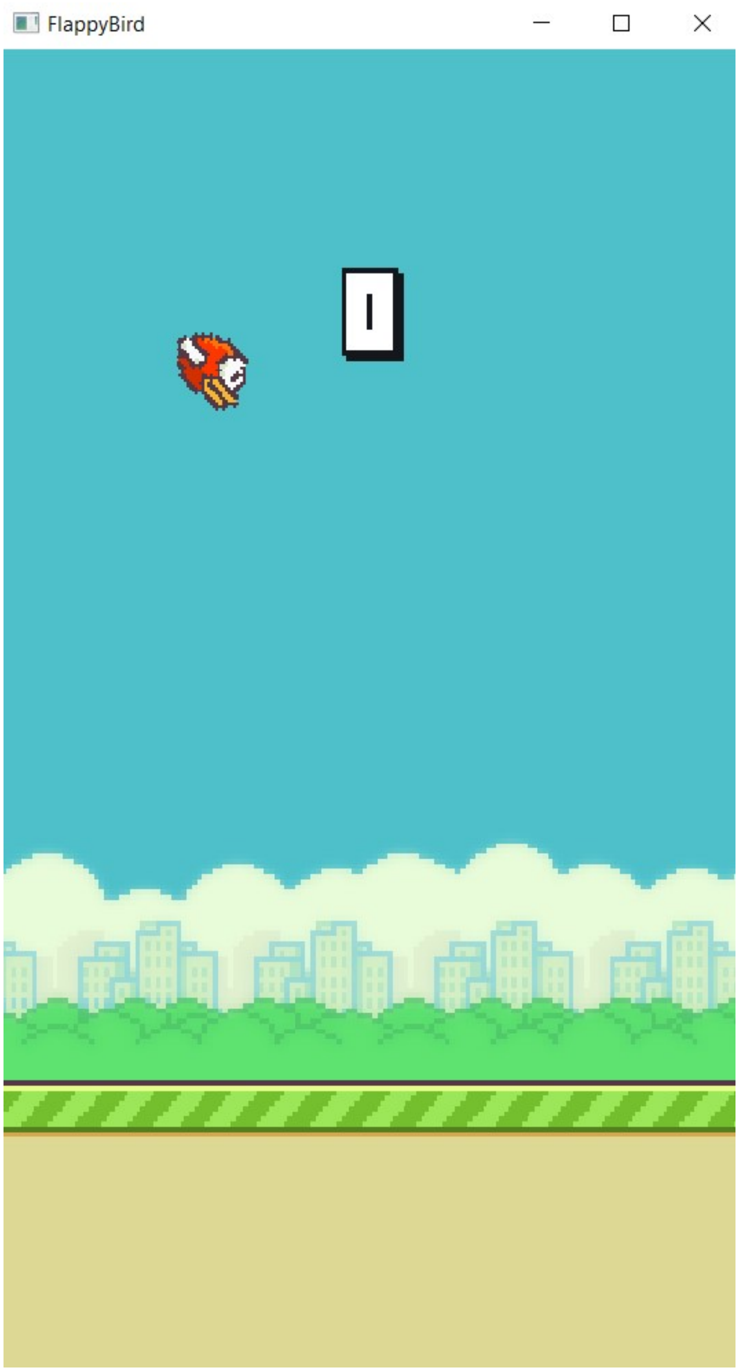
This window contains a background, the name of the game, the image of the bird, and three buttons ("ABOUT", "PLAY", "QUIT"). *The interface's icons, buttons, backgrounds, ... we all have them as png images, we got all these ressources from this link "https://www.kindpng.com/free/flappy-bird/"*

- "Second Interface (GAMEREADY)": This window apperears once we click on the button "PLAY"



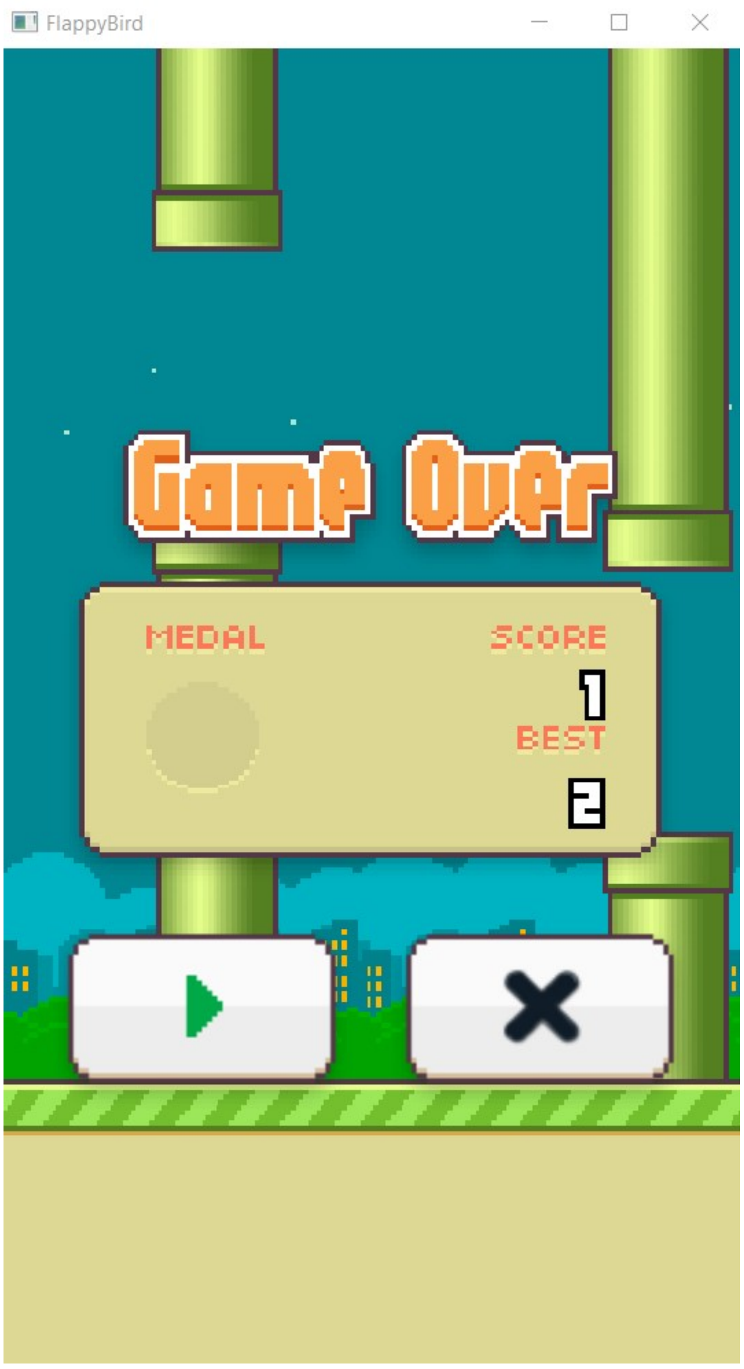
We ramark in this window that we have the same background (in the first capture we have *LightBackground* but in this one we have *DarkBackground*, because each time we lose or we restart the game, the background changes), we have the same remark concerning the bird (but here we will use three colours: BlueBird, YellowBird, RedBird), and we have here a number that increments, as we have also the "Get Ready!", and an image that shows how the game should be played. We have all the previous elements as PNG images.

- "Third Interface (GAMEPLAY)": This interface appears once we click on anything on the screen of the game



In this one we keep just the bird, the record (the incrementing number), and the backgroung image.

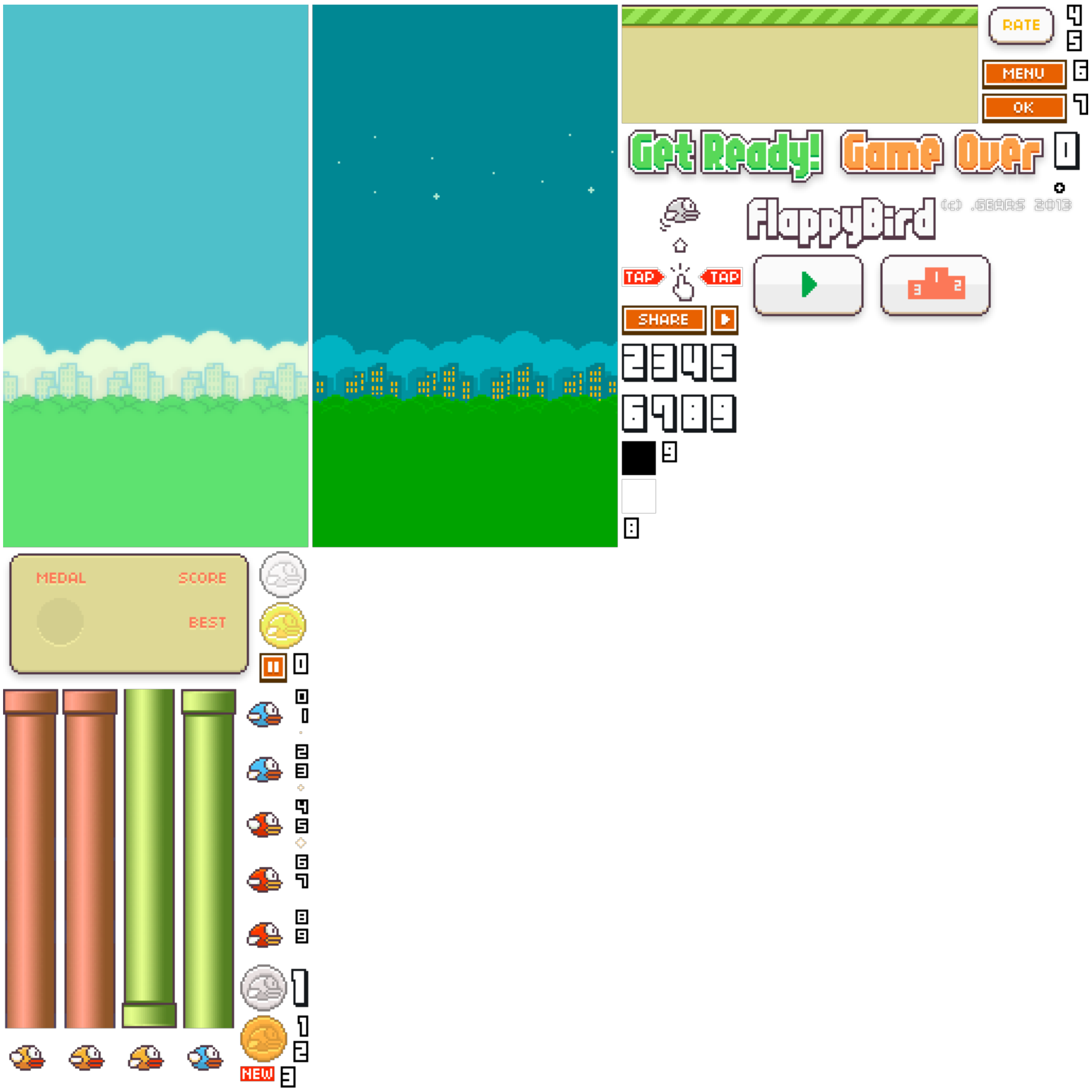
- "Fourth interface (GAMEOVER)": When the player loose, we need to show the player that he lost, so we need to inform him with an image "Game Over", and with a table that gives more informations, (his score, his best-record, and the medal he earned), underneath this table there's the two buttons (PLAY, QUIT).



## 2. Ressources:

### 2.1 Images Ressources:

In the file that contains the code of the game, we have created a file of ressources, our ressources are images and sound effects. For the images: all the icons, buttons, numbers, ... that we have used, we took them all from this website "https://www.kindpng.com/free/flappy-bird/" We gathered all the images in one image so it is easy to check on them

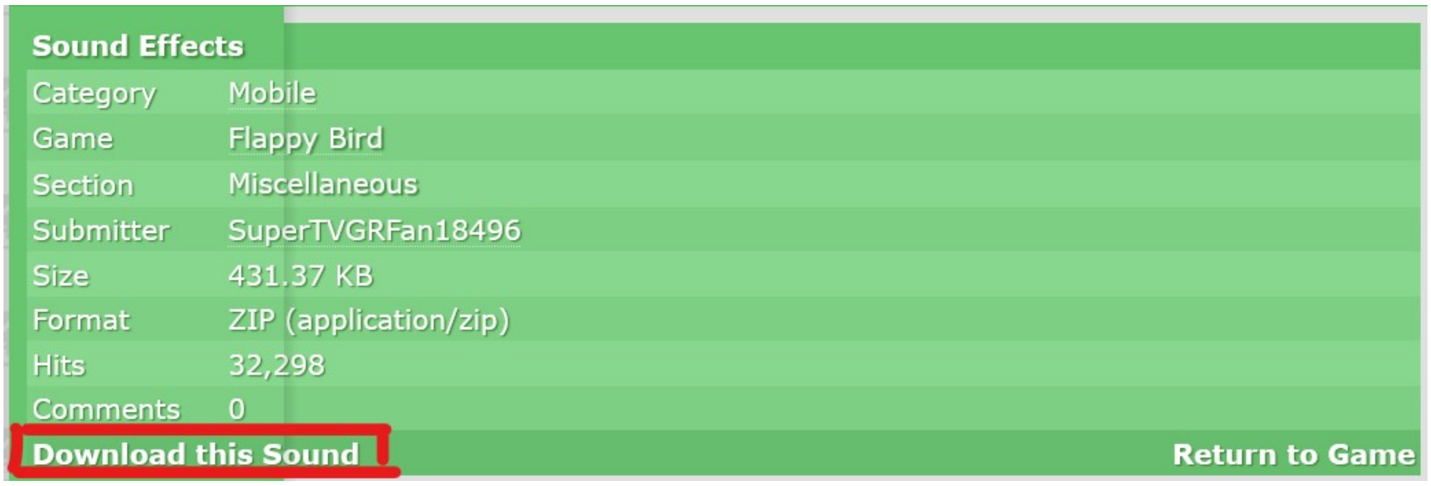


In details, and as shown in the prerequisites section:

- We used two backgrounds (Day and Dark backgrounds);
- 3 birds (blue, yellow, and red) but we remark that we have in this capture 9 birds, we have 3 birds in each similar colour, because during the gameplay, the bird moves his wings in three positions (up , down, or middle) and that's why we have used the 3 birds in each colour;
- We have also, a part from the background that keeps moving;
- We have two pipes with two different positions;
- We see that there are numbers from 0 to 9, those numbers are useful concerning the record, because the record of the player is incrementing;
- We have the buttons, medals, labels, ... .

## 2.1 Sounds Ressources:

We used sound effects so that the game seem a bit closer to the original one, and to amuse the player while playing it. We got the sound effects from this website "https://www.sounds-resource.com/mobile/flappybird/sound/5309/"


| | |
|---|---|
| Sound Effects | |
| Category | Mobile |
| Game | Flappy Bird |
| Section | Miscellaneous |
| Submitter | SuperTVGRFan18496 |
| Size | 431.37 KB |
| Format | ZIP (application/zip) |
| Hits | 32,298 |
| Comments | 0 |
| Download this Sound | Return to Game |

The file we downloaded contains 5 sounds pieces: - Die SFX: This one is played when the bird falls; - Hit SFX: This one is played when the bird hits something, so this one is played simultaniously with the prvious one (Die SFX); - Point SFX: This one is played each time the player earn a point; - Swooshing SFX: This one is played whe the player hits the "Play" button; - Wing SFX: This sound is playedeach time the player press the mouse to move the bird, wih each movement of the wing.

## 3. Coding:

Now we will move to the most important part of this project, the part of proramming this game, so we will need to put all those prerequisites, and give them a role in our program. As we have said previously, we have four interfaces, so we will explain the highlights of the process, because covering each detail will make it very long, and taking into concideration that we will put the file containing all the code.

First things first, we created a new "qmake" "Qt Widget Application", with a "QMainWindow" base class.

We will start with the "First Interface" (the one that contains the name of the game and the three buttons, ...) Well, - Before starting the code, we will need to include some important libraries, we will try to use the ones that we have already studied during this and the previous year: **"QTimer"**: To create repetitive timer, delay to control mouvements, ... ; **"QFile"**: Provides an interface for reading from and writing to files; **"QTextStream"**: So we can conveniently read and write words, lines and numbers. For generating text, QTextStream supports formatting options for field padding and alignment, and formatting of numbers; **"QObject"**: Is the base class of all Qt objects, the central feature in this model is a very powerful mechanism for seamless object communication called signals and slots. We can connect a signal to a slot with connect() and destroy the connection with disconnect(); **"QVector"**: This simply provides a dynamic array; **"QPainter"**: This class performs low-level painting on widgets and other paint devices, it provides highly optimized functions to do most of the drawing GUI programs require. It can draw everything from simple lines to complex shapes like pies and chords. **"QPixmap"**: this class is an off-screen image representation that can be used as a paint device. **"QSound"**: it simply provides a method to play .wav sound files. **"QMouseEvent"**: class contains parameters that describe a mouse event (click or scroll ...). ⇒ We have mentioned above approximateley all the classes we will need, with a simple definition of its role, we have already seen most of them in class, so it will be kind of easy to use them.

- We should put backgrounds and the other stuff, to do that we have to add a "Qt Ressource file" in which we put our ressources (the images).

### 3.1 Coding "MainWindow.h":

In this file we have included the classes mentioned above, the same for the "header files", so the **"MainWindow.h"** contains:

```
enum  GameStatus{GAMETITLE,GAMEREADY,GAMEPLAY,GAMEOVER};
```

⇒ As we have mentioned, those are our four interfaces.

```
 QPushButton  *startButton;
QPushButton  *closeButton;
QPushButton  *infoButton;
```

⇒ Buttons, on the First Interface.

```
 EleBackground  *background; //The background image
EleGround  *ground;         //The lower part of the background (mobile part)
EleBird  *bird;             //The bird (player)
ElePipe  *pipe[3];          //The pipe as table, we have 3 pipes
EleScoreBoard  *scoreboard;
EleReadyBoard  *readyboard;
EleOverBoard  *overboard;
EleTitleBoard  *titleboard;
```

⇒ Elements that we need to draw.

```
 QSound  *soundDie;
QSound  *soundHit;
QSound  *soundPoint;
QSound  *soundSwooshing;
QSound  *soundWing;
```

⇒ We have already explained this part of sounds effects and how we got them, and we will see in the coming lines how we implemented the sound and make them work

```
 int  score;              // The score of the player
QRectF  impactBirdRect;  // The rectangle used to test the impact of the bird
GameStatus  status;      // Mentioned above
QTimer  timer;           // Main  timer,  refresh  the  interface.
```

⇒ The declaration of timer, status, ...

```
 void  gameTitle();
void  gameReady();
void  gamePlay();
void  gameOver();
```

⇒ Corresponding to 4 game status or the four interfaces.

```
 void  paintEvent(QPaintEvent  *);       //contains event parameters for paint events
void  mousePressEvent(QMouseEvent  *);  //contains event parameters for mouse events
void  resizeEvent(QResizeEvent  *);     //contains event parameters for resize events that are sent to widgets that have been resized
```

⇒ The declaration of events, we declare them as public.

```
 void  getScore();     // To get the score that increases.
void  startGame();    // The (Start) button clicked, the game began.
void  closeGame();    // The (Quit) button  clicked, the game exit.
void  displayInfo();  // The  button  that  shows  informations
void  setButtonVisible(bool,bool,bool);  // set the 3 buttons above visible
```

⇒ The declaration of slots, we declare themas public slots.

## 3.2 Coding "MainWindow.cpp":

In this file (MainWindow.cpp), we develop all the declarations in the (MainWindow.h) file, the first thing we did in this file is to include the header file.

```
 MainWindow::MainWindow(QWidget  *parent)
:  QMainWindow(parent)
{
    qsrand(QTime::currentTime().second());
    this->bufferPixmap  =  new  QPixmap(288,512);
    this->resize(540,960);
```

⇒ We set the size.

### StartGame Slot:

```
 this->startButton = new QPushButton(this);
// We create the button
this->startButton->setGeometry(QRect((20.0/288)*this->width(),
                              (341.0/512)*this->height(),
                              (117.0/288)*this->width(),
                              (71.0/512)*this->height()));
// We set the geometry of the button height,width)
this->startButton->setStyleSheet("QPushButton{border-image:url(:/image/image/button_play.png);}"
                                 "QPushButton:pressed{margin: 2px 2px 2px 2px;}");
// We set the background of the image
connect(this->startButton,SIGNAL(clicked()),this,SLOT(startGame()));
// We give the convenient action related to pressing this button
```

⇒ We will show in the next lines what we will do about  **"startgame"** slot, because when we click on **"startgame"** we should play the game, so:

```
 void  MainWindow::startGame()
{
    // We initialise (init) all the game elements.
    this->background->init();
    // background
    this->ground->init();
    // Lower part of the background
    this->bird->init();
    // The dodging bird
    this->scoreboard->init();
    // The score number that will increase
    this->readyboard->init();
    // The "GetReady!" label
    this->overboard->init();
    // The small tutorial board
    this->pipe[0]->init();
    this->pipe[1]->init();
    this->pipe[2]->init();
    // The pipes
    // Game  Start.
    this->gameReady();
    this->soundSwooshing->play();
    // Starting the sound by using "play()"
}
```

⇒ We remark that we have set the button and then we assigned, the related actions.

### CloseGame Slot:

```
 this->closeButton = new QPushButton(this);
// We create the Close button
this->closeButton->setGeometry(QRect((151.0/288)*this->width(),
                              (341.0/512)*this->height(),
                              (117.0/288)*this->width(),
                              (71.0/512)*this->height()));
this->closeButton->setStyleSheet("QPushButton{border-image:url(:/image/image/button_close.png);}"
                                 "QPushButton:pressed{margin: 2px 2px 2px 2px;}");
connect(this->closeButton,SIGNAL(clicked()),this,SLOT(closeGame()));
```

⇒ Well, the exception here is that, if we click on the**"closeGame"** we should leave the game, so the action will be easy. ⇒ And to make it happen:

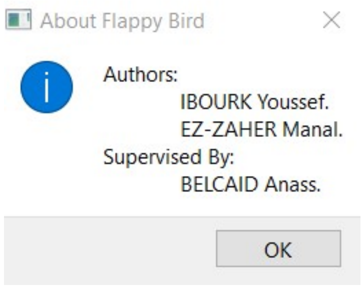
```
 void  MainWindow::closeGame()
{
    this->soundSwooshing->play();
    // We start the "swooshing sound"
    this->close();
    // And then easily we close the game
}
```

And regarding the First Interface, the last button is the **"Info Button"**

```
   this->infoButton  =  new  QPushButton(this);
  //We create the button
  this->infoButton->setGeometry(QRect((106.5/288)*this->width(),
                              (300.0/512)*this->height(),
                              (75.0/288)*this->width(),
                              (48.0/512)*this->height()));
  //We set the button's dimensions
  this->infoButton->setStyleSheet("QPushButton{border-image:url(:/image/image/button_rate.png);}"
                                 "QPushButton:pressed{margin: 2px 2px 2px 2px;}");
  //We upload the background of the image
  connect(this->infoButton,SIGNAL(clicked()),this,SLOT(displayInfo()));
  // To relate the button with the signall that we will assign to it
```

⇒ The signal relate to this button is to pop up a window to tell informations about the creators or other informations ⇒ So :

```
 void  MainWindow::displayInfo()
{
     this->soundSwooshing->play();
     // We we hit this button the "swooshing sound" we be played
     QMessageBox::information(this,"About  Flappy  Bird","Authors:\n\tIBOURK  Youssef.\n\tEZ-ZAHER  Manal.\nSupervised  By:\n\tBELCAID  Anass.");
     //Here we set the title of this window "About Flappy Bird"
     //we write the content of this window: authors ....
 }
```



About Flappy Bird

Authors:
IBOURK Youssef.
EZ-ZAHER Manal.
Supervised By:
BELCAID Anass.

⇒ So the output of this button will be as follow: And now we go to the initialisation of the game elements

```
 this->background  =  new  EleBackground();
this->ground  =  new  EleGround();
this->bird  =  new  EleBird();
this->scoreboard  =  new  EleScoreBoard();
this->readyboard  =  new  EleReadyBoard();
this->overboard  =  new  EleOverBoard();
this->titleboard  =  new  EleTitleBoard();
this->pipe[0]  =  new  ElePipe(0);
this->pipe[1]  =  new  ElePipe(1);
this->pipe[2]  =  new  ElePipe(2);
```

After that, we go to initialise the sound effects with their container file

```
 this->soundDie  =  new  QSound(":/sounds/sounds/sfx_die.wav");
this->soundHit  =  new  QSound(":/sounds/sounds/sfx_hit.wav");
this->soundPoint  =  new  QSound(":/sounds/sounds/sfx_point.wav");
this->soundSwooshing  =  new  QSound(":/sounds/sounds/sfx_swooshing.wav");
this->soundWing  =  new  QSound(":/sounds/sounds/sfx_wing.wav");
```

Next, we set a timer as a refresh rate of 50 HZ

```
 connect(&timer,SIGNAL(timeout()),this,SLOT(update()));
timer.start(20);
```

Next, we need to define a function that each time the player passes in between the two pipes, the score should increments

```
 connect(this->pipe[0],SIGNAL(pipePass()),this,SLOT(getScore()));
connect(this->pipe[1],SIGNAL(pipePass()),this,SLOT(getScore()));
connect(this->pipe[2],SIGNAL(pipePass()),this,SLOT(getScore()));
```

⇒ This function is **"pipepass"**

Now we move to one of the most important parts of the code (the mouse press event)

```
void  MainWindow::mousePressEvent(QMouseEvent  *event)
    {
    //When the game starts the bird is up
    if(  this->status  ==  GAMEPLAY  &&  event->button()  ==  Qt::LeftButton  )
    //The button we will play with is the left button of the mouse
        {
        this->soundWing->stop();
        //we stop the sound
        this->bird->birdUp();
        //the bird is up
        this->soundWing->play();
        //and them we play the sound effect
        }
        /*When  the  game  is  ready  ,  start  the  game.  */
        if(  this->status  ==  GAMEREADY  &&  event->button()  ==  Qt::LeftButton  )
        {
            this->gamePlay();
        }
    }
```

Now we move to define **"paintevent()"**

```
 void  MainWindow::paintEvent(QPaintEvent  *)
{
QPainter  painter(this->bufferPixmap);
// We Draw  all  the  elements.
this->background->logic();
this->background->draw(&painter);
this->pipe[0]->logic();
this->pipe[0]->draw(&painter);
this->pipe[1]->logic();
this->pipe[1]->draw(&painter);
this->pipe[2]->logic();
this->pipe[2]->draw(&painter);
this->bird->logic();
this->bird->draw(&painter);
this->scoreboard->logic();
this->scoreboard->draw(&painter);
this->readyboard->logic();
this->readyboard->draw(&painter);
this->titleboard->logic();
this->titleboard->draw(&painter);
this->ground->logic();
this->ground->draw(&painter);
this->overboard->logic();
this->overboard->draw(&painter);
QPainter  mainWindowPainter(this);

mainWindowPainter.drawPixmap(QRect(0,0,this->width(),this->height()),*this->bufferPixmap);
if(this->status  ==  GAMEPLAY)
{
// We update  the  impact  rectangle  of  bird.
this->impactBirdRect.moveCenter(this->bird->getBindRect().center());
// We To  test  if  the  impact  happened.
if(this->impactBirdRect.intersects(this->ground->getBindRect()))
{
this->soundHit->play();
this->gameOver();  //Game  over
}
if(this->impactBirdRect.intersects(this->pipe[0]->getRect(above))
||this->impactBirdRect.intersects(this->pipe[0]->getRect(following))
||this->impactBirdRect.intersects(this->pipe[1]->getRect(above))
||this->impactBirdRect.intersects(this->pipe[1]->getRect(following))
||this->impactBirdRect.intersects(this->pipe[2]->getRect(above))
||this->impactBirdRect.intersects(this->pipe[2]->getRect(following)))
{
this->soundHit->play();
this->soundDie->play();
this->gameOver();  //Game  over
}
}
```

In this paragraph, we will make a process of enabling/disabling the furnitures: So in **"gameTitle"** part or (Fiorst Interface)

```
 void  MainWindow::gameTitle()
{
    this->background->enabledLogic  =  true;
    this->background->enabledDraw  =  true;
    this->titleboard->enabledLogic  =  true;
    this->titleboard->enabledDraw  =  true;
    this->bird->enabledLogic  =  false;
    this->bird->enabledDraw  =  false;
    this->ground->enabledLogic  =  true;
    this->ground->enabledDraw  =  true;
    this->overboard->enabledLogic  =  false;
    this->overboard->enabledDraw  =  false;
    this->pipe[0]->enabledLogic  =  false;
    this->pipe[0]->enabledDraw  =  false;
    this->pipe[1]->enabledLogic  =  false;
    this->pipe[1]->enabledDraw  =  false;
    this->pipe[2]->enabledLogic  =  false;
    this->pipe[2]->enabledDraw  =  false;
    this->readyboard->enabledLogic  =  false;
    this->readyboard->enabledDraw  =  false;
    this->scoreboard->enabledLogic  =  false;
    this->scoreboard->enabledDraw  =  false;
    this->setButtonVisible(true,true,true);
    this->status  =  GAMETITLE;
}
```

⇒ We remark that in each feature, we have an **"enabledLogic"** and **"enabledDraw"** ⇒ **"enabledLogic"** and **"enabledDraw"** are boolean variables to mark the presence of each fourniture ⇒ **"true"** for present and the contrary for **"false"**

That's all we need to define in the **"MainWindow.cpp"** file The other header files and ".cpp" are a little traditional implementation, that we have treatedits concept during the class, and during the projects we were asked to realise.

## Conclusion:

Honestly, this project is one of the most intense projects we have realised, especially, that it was our choice to choose the game, this gave us the opportunity to choose a game that we played when we were young and that we miss, especially that the owner banned it from the **"IOS"** and **"ANDROID"** devices, and that was a sufficient reason to choose this game We have found many barriers during the programmation, the biggest one is the experience, we have a little amount of experience, so we took it as an advantage to be motivated and pushed to make this game happen, we took it as an advantage to make many researches. Before the code succeded we have made a countless amount of attempts that all failed. And finally, all the courses we took, and informations we got from a very good Professor M. BELCAID was sufficient, and ofcourse with some researches.