FLAPPY BIRD CONTROLLER THROUGH MATLAB

Yulun Zhuang

Southern University of Science and Technology, Shenzhen, China

ABSTRACT

Flappy Bird is a mobile game[1] that involves tapping the screen to navigate a bird through a gap between pairs of vertical pipes. When the bird passes through the gap, the score increments by one and the game ends when the bird hits the floor or a pipe. Surprisingly, Flappy Bird is a very difficult game and scores in single digits are not uncommon even after extensive practice. The aim of this project is to develop an interactive controller app through MATLAB app designer to play the game autonomously. The control algorithm is implemented by bang-bang control and PID control. Besides, all controller's parameters can be tuned in app's interface and the trajectory of bird and tubes can be drawn in real time. In order to show the effect of tuned parameters, the bird and tubes are represented by circle and rectangle respectively in the inserted figure.

Keywords: Flappy Bird, Controls, MATLAB App

1. INTRODUCTION

The game Flappy Bird is a side-scrolling mobile game, which was a very popular in early 2014. The game itself is repetitive, hard and addictive. The objective is to direct a flying bird which moves continuously to the right between sets of pipes. If the bird touches the pipes, the game ends. Each time the player taps the screen, the bird briefly flaps upwards; if the screen is not tapped, the bird falls to the ground due to gravity, which also ends the game.

Originally this game was released for mobile phones, however a different developer implemented it in MATLAB and released it on GitHub[2]. During the development the MATLAB version was modified to accommodate the control system and to visualize the information.

The aim of this project is to develop an interactive controller app through MATLAB app designer to play the game autonomously. The control algorithm is implemented by two kinds of controllers: (1) bang-bang control tuned by two boundary conditions and (2) PID control tuned by heuristic methods. Besides, all controller's parameters can be tuned in app's interface and the trajectory of bird and tubes can be drawn in real time. In order to show the effect of tuned parameters, the bird and tubes are represented by circle and rectangle respectively in the inserted figure.

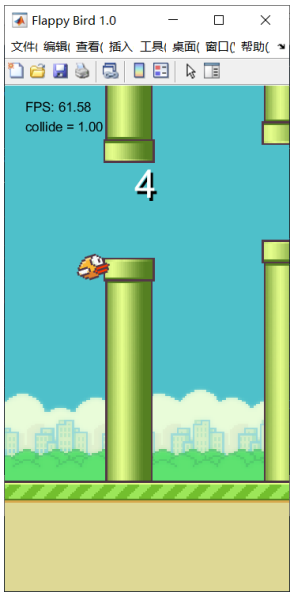

FIGURE 1: Flappy Bird Game by MATLAB

2. CONTROLLER DESIGN

To start the design of controller, I first figure out the game logical and some basic game parameters. The source code is exported from GUIDE with more than 500 lines. Luckily the game is implemented by modules, so I read and test some of them and finally got the right place for my codes.

2.1 System modeling

The dynamics equations of the bird in y direction can be written as follow:

$$v_{i+1} = \begin{cases} v_i - g, & key\ disable \\ 2.5, & key\ enable \end{cases} \quad (1)$$

$$y_{i+1} = y_i + v_i - \frac{1}{2}g \quad (2)$$

where i is the time index, y_{i+1} is the height of the middle of the bird at time i+1 (the y-axis points downward), v_{i+1} is the bird's velocity at time i+1 and the gravity is 0.15. Key enable indicates bird flaps to move up and key disable indicates that the bird does nothing. Note that y_{i+1} and v_{i+1} is in unit of pixels and pixels per unit time, respectively. The time step is 1 unit so that time does not appear in any equation.

The bird and tubes are operating in two independent coordinate systems, the transform between them is:

$$y_{tube} = 176 - y_{bird} \quad (3)$$

where y is in pixel. The coordinates representing bird are fixed in x direction but fluctuating in y direction, while those representing tubes are scrolling in x direction but fixed in y direction. Especially, the two coordinate systems are opposite in y direction.

There are only three tubes processed at a time, but the index of the front tube goes from 1 to 3. The y coordinate of the front tube can be determined by

```
setPointX = mod(Tubes.FrontP,3)+1;
TubePosY = Tubes.VOffset(setPointX);
```

where the tubes are moving in the horizontal direction with constant speed of 1 pixel per unit time. The horizontal distance between tubes and the vertical gap between tubes are fixed while the location of gap is generated randomly.

2.2 Bang-bang controller

A bang–bang controller, also known as a hysteresis controller, is a feedback controller that switches abruptly between two states[3]. Since the y-coordinate of bird and the front tube are determined in the same coordinate system, a bang-bang controller can be set up easily by comparing the sign of the displacement between bird and the front tube.

```
BirdPosY = 176 - Bird.ScreenPos(2);
setPointY = TubePosY + c;
if (BirdPos <= TubePos)
    Bird.SpeedY = -2.5;
end
```

where the *setPointY* is the first boundary condition, if the displacement is positive the switch is off, if the displacement is negative, then the switch is on and the bird will have a 2.5 upper speed. The higher the *setPointY*, the easier the bird will hit the upper tube, while the lower the *setPointY*, the easier the bird will hit the lower tube[4].

The second boundary condition *setPointX* is the distance where the bird passes the last tube, which decides when to refresh *setPointY* to the next one.

The performance of bang-bang controller will be discussed later.

```
if Tubes.ScreenX(Tubes.FrontP) +...
    app.xOffset < Bird.ScreenPos(1)
    setPointX = mod((Tubes.FrontP),3)+1;
end
```

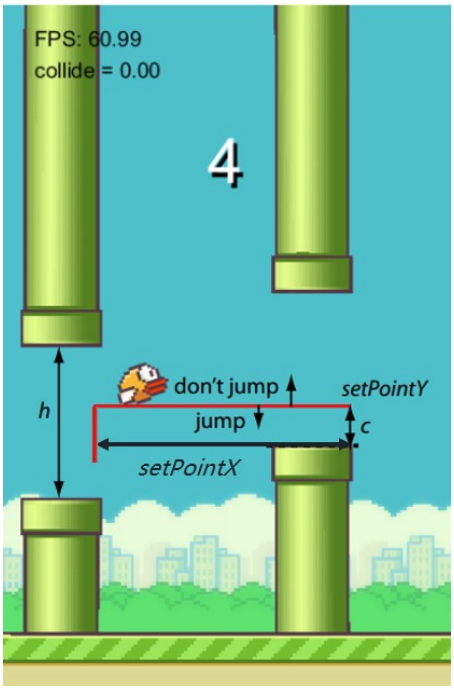

FIGURE 2: Two Boundary Setpoints

2.3 PID controller

The implementation process of PID algorithm is very simple, that is, the feedback is used to detect the deviation signal, and the deviation signal is used to control the controlled quantity. The controller itself is the sum of proportion, integration and differentiation. The PID controller I have learnt in Control Engineering course is in frequency domain:

$$U(s) = K_p + \frac{K_i}{s} + sK_d \tag{4}$$

where K_p is proportional constant, K_i is integral constant and K_d is derivative constant. However, to control the flappy bird in real time, I need operating the PID controller in time domain:

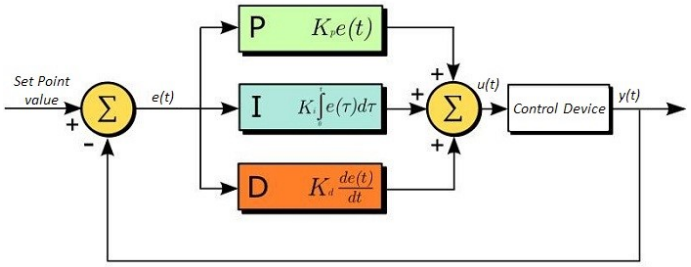

FIGURE 3: PID Controller in Time Domain

With the basic principle of PID algorithm, it must be decentralized in order to achieve on computer. We assume that the sampling period of the system is T, the discrete form of PID at the *k*th sampling period is expressed as:

$$U(k) = K_p e(k) + K_i \sum e(k) + K_d(e(k) - e(k-1)) \quad (5)$$

where $e(k)$ is taken as the displacement between *BirdPosY* and *setPointY* at the *k*th sampling period. The PID algorithm is implemented with four member variables:

```
Kp = 1;
Ki = 0;
Kd = 0;
LastErr = 0;
IntegralErr = 0;
```

then the controlling process:

```
err = BirdPos - TubePos;
app.IntegralErr = app.IntegralErr + err;
PIDresult = app.Kp*err +…
    app.Ki*app.IntegralErr +…
    app.Kd*(err-app.LastErr);
app.LastErr = err;
```

This is the realization of a simple position PID controller, of course, without considering any interference conditions, just to the mathematical formula of the computer language. The performance of PID controller will be discussed later.

3. GUI DESIGN

The graphical user interface is implemented by MATLAB App Designer, which is very powerful and convenient for simple app designing.

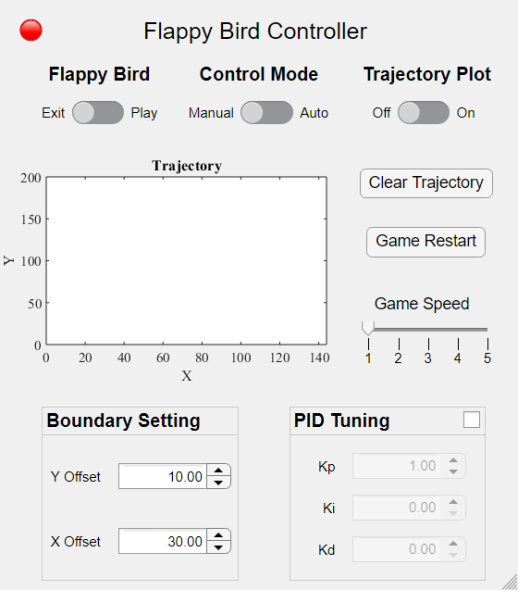

FIGURE 3: Controller's Interface

3.1 Interface and components

The whole interface is shown in Figure 4. This interface is combined with three parts, the mode switch section, the plotting section and the parameter tuning section. And the three parts are implemented by several simple elements shown as follow:

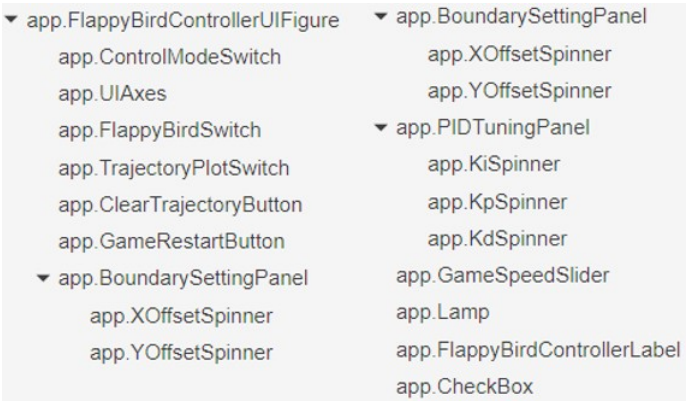

FIGURE 4: Controller's Components

3.2 App communication

The open source Flappy Bird is an m file covered by a single main function to hide its parameters. In order to pass app's variables into the main game, I add *app* as an input parameter to the main function, then all the properties declared in app can be call in game by *app.propertiy* as well as all call back functions.

```
function flappybird (app)
```

It should be noted that all the function related with plotting should add the *app.UIAxes* as the first input parameter.

3.3 Trajectory graph

When the game is running, the bird and tubes can be plotted in the trajectory graph section in real time. The main game is set to running at 60 FPS, while the plotting is at 12 FPS. If the plotting frequency is the same as the game, it will cause a huge delay on both windows. Plot at 12 FPS is a proper frequency that will cause little affect for game fluency and the graph can be plotted continuously.

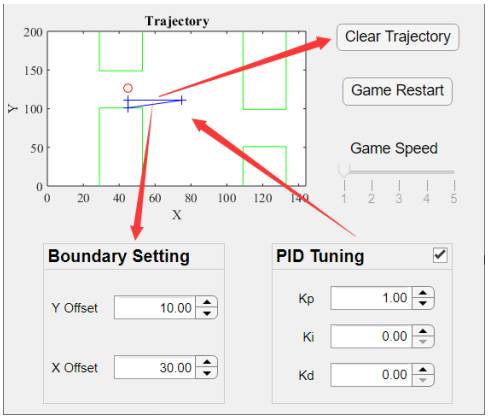

FIGURE 5: Logical of Main Panels

where the bird is represented by a red circle, tubes are represented by green rectangle and the two boundaries are represented by a blue triangle.

The main purpose of this graph is to view the changing in bird's behavior directly and clearly when tuning controllers' parameters.

3.4 Parameter tuning

When the game is running, there are six parameters can be tuned. The first two is the boundary setting, followed by K_p, K_i and K_d. All altering of them can be applied immediately.

The last one of them is the game speed which can only be changed when bird hits a tube or fall to the ground, because in the main loop of game, the FPS is calculating due to the change of each frame. If the refresh rate of frame is changed during flying, the game will crash somehow.

4. RESULTS AND DISCUSSION

In order to test the performance of bang-bang controller and PID controller, I simulated 10 games and each game was initialized with a random number generator. This ensures that no same set of pipes and gaps are present for the given simulation run.

Table 1 shows results for the bang-bang controller with heuristic tuning of boundary parameters (Sec. 2.2). It is only able to score an average score of 56.6 points per game. While this is significantly higher than what we were able to do by playing the game manually (roughly 0– 6 points per game).

Run Turn	Score
1	9
2	30
3	19
4	105
5	29
6	41
7	40
8	135
9	135
10	23

TABLE 1: Scores for Bang-bang Controller

However, the PID controller can hardly lose, since it changes the rules of original game to some extent. One tap can only give a speed of 2.5, but not any other values. Thus I limit the compensated output to positive, which means the PID can only pull the bird upward, in order to make the bird's behavior more closer to the original one.

REFERENCES

[1] Flappy bird. https://en.wikipedia.org/wiki/Flappy_Bird. Accessed: April 20, 2020.

[2] Pranav Bhounsule (2020). pab47/FlappyBirdController (https://www.github.com/pab47/FlappyBirdController), GitHub. Retrieved May 24, 2020.

[3] Shu, Y., Sun, L., Yan, M., and Zhu, Z., 2014. Obstacles avoidance with machine learning control methods in flappy birds setting. Department of Mechanical Engineering, Stanford University.

[4] Matthew Piper, Pranav A. Bhounsule, and Krystel K. Castillo-Villar., 2017. How to beat flappy bird: A mixed-integer model predictive control approach. The University of Texas at San Antonio.