Term Project Report

B09901043 陳昱樺
B09901134 林容丞

• Topic : Crazy Kitchen
• GitHub link : https://github.com/Chen-Yuhua/Final-Project
• Demo video : https://youtu.be/qfzO6oCcIFI

• Introduction

Crazy kitchen is an interesting game for two players. First, players need to collect ingredients such as bread, meat, and tomato, and carry them to the kitchen. Second, players should start processing these ingredients to make a hamburger. Finally, players need to carry the hamburger to the serving area. When a hamburger is served, players will get one point.

There are two modes in Crazy Kitchen: versus mode and team mode. In versus mode, each player will make his own dishes individually. Player with the higher score will win the game. In team mode, two players will cooperate with each other to complete as many dishes as they can. The higher the score is, the stronger friendship they have.

The players have 120 seconds to finish their dishes. If someone is tired, he can press the pausing button to pause the game. Once the pause button is pressed again, the game will continue. Besides, there is cheerful music playing during the game, so players can enjoy in Crazy Kitchen and acquire oceans of joy.

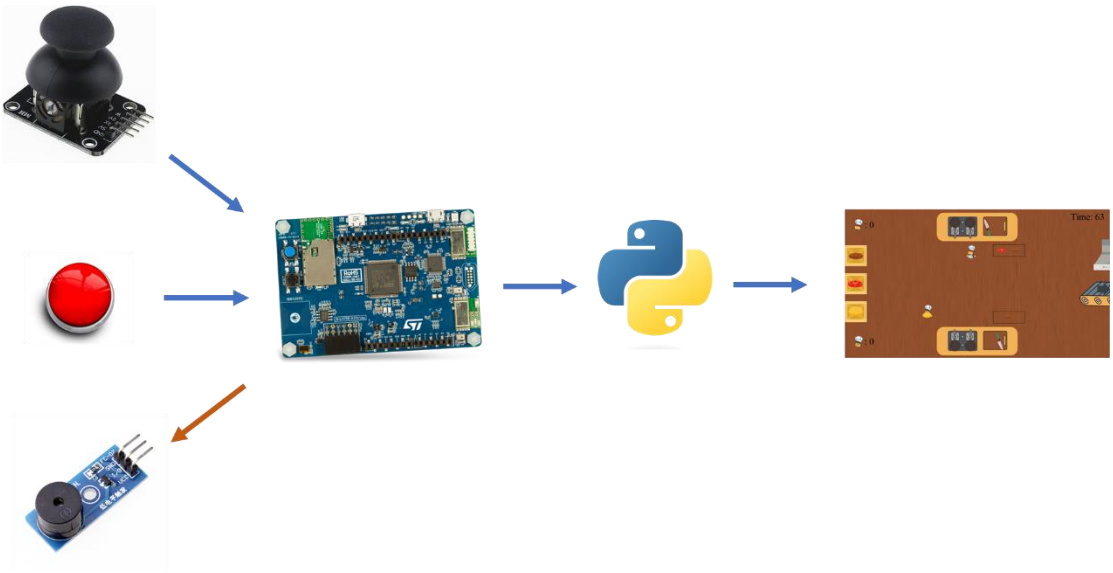
• Motivation

While playing the game Overcooked, we are not satisfied by the variety of dishes and characters in the game, leading to a desire to modify certain elements. Moreover, taking advantage of the knowledge and techniques acquired in class, we believe that developing our own version of a cooking game is an achievable goal. Therefore, for this term project, we have decided to design and create a unique cooking game that offers players an enjoyable experience.

• Method

The following graph shows the architecture of our project. There are two kinds of

input devices, a joystick and a button, which are in charge of data collection. A buzzer plays the role of output device that plays the music. These devices are all connected to the STM32 board. Then, the STM32 board will connect to a personal computer that runs the python program using Wi-Fi. Eventually, the python program will generate a window showing the scene for the game.



• Design description

1. Data collection

There are five input devices for this project: two joysticks and three buttons. Each joystick represents a player, and we collect its displacement in x-axis and y-axis in an analog way. The resolution for each dimension is 100000. As for the three buttons, two are for the two players, and the other one is a pausing button. We detect whether a button is pressed in a digital way. To make the game fluent, we collect input data once every 10 milliseconds.

2. Data processing

After collecting input data, we will do data processing to convert them into useful information for further usage.

For data about joysticks, we set some boundaries to handle wrong input values. If the input sensor value is valid, the program will detect the direction in which a player tends to move his character. Then, a variable will record this direction for further use. That is, we adopt a discrete method to process joystick data.

For data regarding buttons, we will calculate the pressing duration to decide if a press attempt is valid. If a press attempt of the pausing button is valid, the game will

pause or continue considering the current condition. For two player buttons, pressing them continuously will do the processing that integrates ingredients into dishes.

3. Communication

Now, we have information about input devices, and we need to transmit them to the computer to update the game scene. The communication method we implement is Wi-Fi. We have finished socket programming in C++ on STM32 board and python on a personal computer.

To send useful information, we use a string whose length is five bytes. The first two bytes will record information about the first player. One is for the moving direction of the character, and the other is for the pressing condition for the player one button. The following two bytes record information about the second player, and the format is similar to the one of the first player. The purpose of the last byte is recording the pressing condition of the pausing button. Moreover, the communication rate is the same as the data collection rate. That is, once for every 10 milliseconds.

4. Music

To make players more comfortable when playing Crazy Kitchen, we have made music for this game. We use a buzzer as the output device to play the music. The technique we use in this part is pulse width modulation.

First, we control the frequency of the buzzer to manage the pitch of a note. Second, we set the pulse width to a suitable value to change the duration of each note. Then, we implement a waiting period for every note to enhance the auditory experience for players.

By the above techniques, we can play the music with a buzzer successfully. Although the quality of the music might not be as good as the one a MP3 module generates, we have implemented the pulse width modulation technique in our project. Also, we have spent time composing a song for Christmas and use it as the final music played in our game.

5. Power saving

When the game is in progress, the music will play continuously. As a result, the buzzer will consume power. With a view to saving energy, we implement a power saving technique regarding music in our project. If a player presses the pausing button to pause the game, the music will stop playing to save power. If the game continues, the music will start playing from the place it stops. In addition, when players are choosing the game mode, the buzzer module will not play the music to prevent extra power being consumed.

6. Tkinter

Taking advantage of the python package tkinter, we can display objects in the window easier. Methods and attributes used are: coords(), find_overlapping(), itemconfig(), create_text(), delete(). Attribute coords() can help achieve the coordinate of a specific object on the canvas. Method find_overlapping() can easily find items that overlap within a specified rectangle range. Itemconfig() method is able to modify the configuration of a certain item. Create_text() draws a text on the canvas. Method delete() enables us to remove unneeded items that we no longer require.

7. Shadow

While moving the character, if we detect collisions after directly moving it, there will be a glitch if it hits an obstacle since we have to move it back. Therefore, as a solution, we create a transparent shadow for the character. While the player attempts to move the character, its shadow will move for the character first, and the collision situation of the shadow is detected by the function find_overlapping(). If it is safe for the character, meaning the shadow does not collide with any obstacle, the character is moved to the desired place. On the other hand, if it is not safe, the shadow is returned to the original position.

As for picking up and placing the ingredients, processing ingredients, and sending dishes, we also create shadow boxes for the ingredient places, the kitchens, and the serving area. After detecting the collision between the character's shadow and the shadow of boxes, we can display the corresponding ingredient in the character's hand, process the food into dishes, or send out the meal and add points up based on the type of the shadow box.

8. Button

For each player's button, we have to detect two different actions: short press and long press. Since we would like the communication between STM32 and python to be continuous, in other words, without delay for determining the actions, we decide to deal with this issue on the python side. Eventually, we classify an action as a long press if a "pressed" signal is received more than 5 times continuously, and view it as a short press otherwise. In order to make this classification work in real time, reactions for short presses shouldn't be performed on the screen before we can ensure it isn't a long press. A counter is also required to add up the received times and reset to 0 when receiving a "release" signal.

9. Game flow

The main algorithm for the game is composed of two game loops. During the

runtime of the game loops, once the python side receives data from the STM32, it extracts information from the message and reacts accordingly. The first one is activated when the game starts, and its main purpose is to receive the signal to choose between the two modes and to catch the “start” signal. In the second game loop, information includes directions to move the two characters, whether the user buttons are pressed, and whether the pausing button is pressed.

• Result

We successfully develop a game with fancy art design for two players. Players can easily control their characters using joysticks, take ingredients or dishes with a short press of the button, and process their ingredients in the kitchen with a long press. In addition, we design two game modes in this project: versus mode and team mode, enhancing the overall variety of the game. Players can pause the game by pressing the pausing button, which not only puts the concept of power saving into practice, but also provides a comfortable user experience. There is also a buzzer playing music during the game, creating an immersive experience for the players. We believe that people can enjoy and have fun in Crazy Kitchen.

• Contribution

1. 陳昱樺: Data collection, data processing, communication, and music.
2. 林容丞: Communication, game flow, game scene, and image design.

• Reference

1. Joystick module usage
https://shop.mirotek.com.tw/arduino/arduino-adv-8/
2. Button module usage
https://blog.jmaker.com.tw/arduino-buttons/
3. Buzzer module usage
https://shop.mirotek.com.tw/arduino/arduino-start-10/
4. Mbed program example for socket programming
https://github.com/ARMmbed/mbed-os-example-sockets
5. Pitches for notes
https://gist.github.com/mikeputnam/2820675
6. Game example using Tkinter
https://github.com/chaiyenwu/python

7. Tkinter Canvas Example
https://shengyu7697.github.io/python-tkinter-canvas/?fbclid=IwAR09GuSAVgkDUkuLKacSlHE6nP6Q70kdlGXi_zEaXvQenwKF8RxeAe3DS7E