Final Presentation

Interactive Dino Run

組員:李亦鎧 黃博聞 陳竣瑋

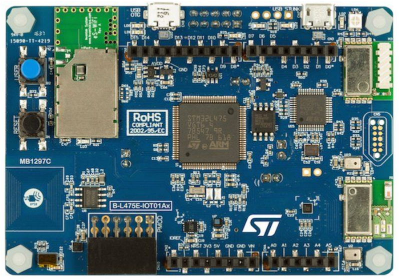
Table of contents

Motivation

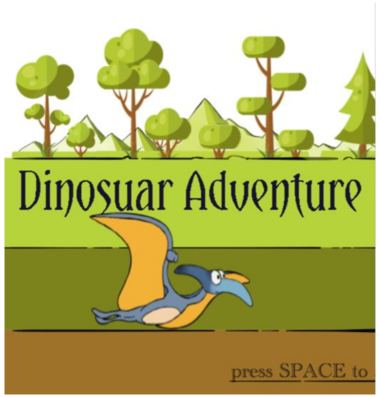
- We used to enjoy playing Google's built-in dinosaur game during offline moments.
- While it was a fun way to kill time when the internet was down, it eventually felt repetitive and dull after playing for a while.
- We decided to create our own version of a custom dinosaur game!

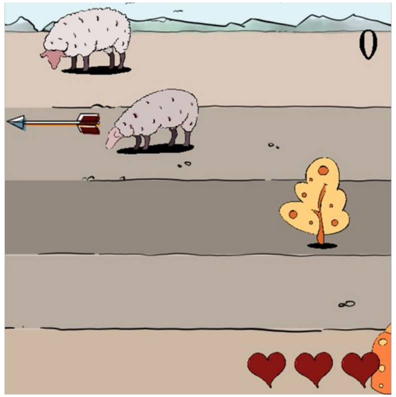
Abstract

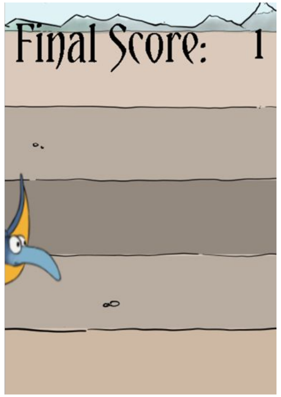
- In this project, we'll use the SDL2 library in C++ to develop a game. Additionally, we'll utilize the STM32CubeIDE to program the B-L475E-IOT01A board, which will serve as the controller for our character and game mechanics.

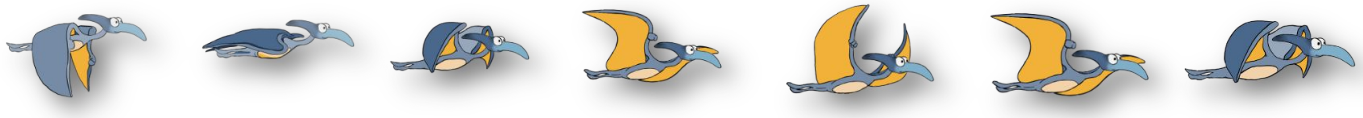


Game Introduction

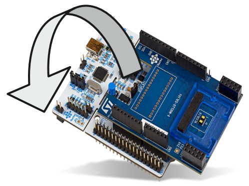


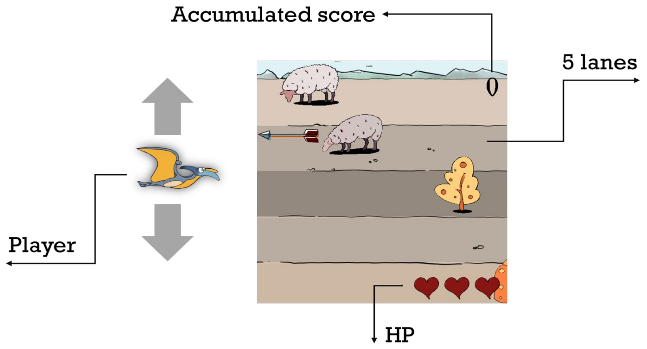


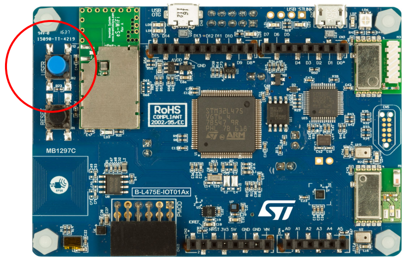




Game Introduction

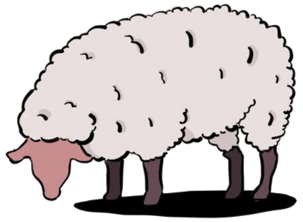
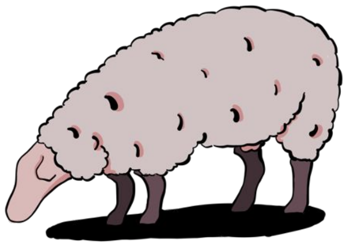
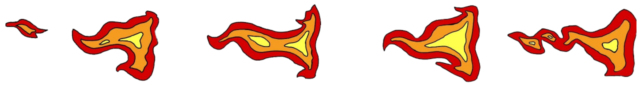


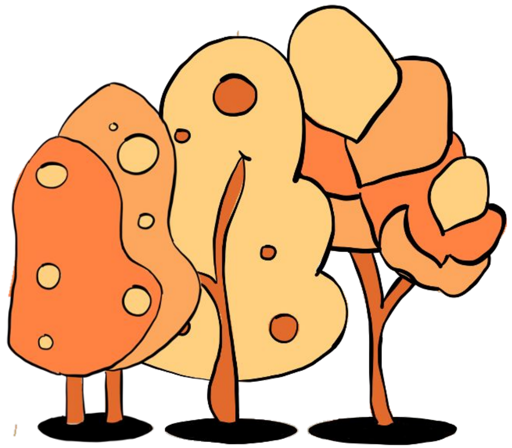




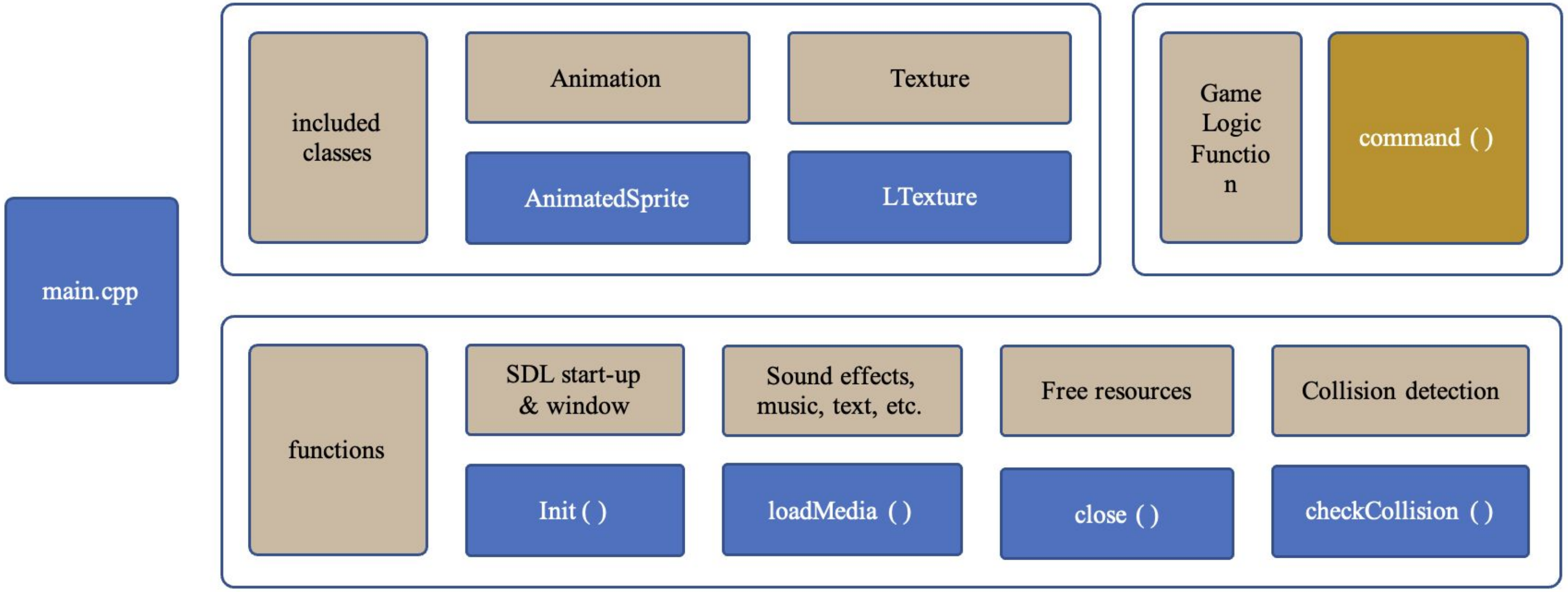
- Push the button to start the game.
- Flip the board to move the character to move up/down.
- Press and hold the button for one second before releasing it to recover health
- Tap it briefly to shoot fire and destroy obstacles.

Game Introduction - Game Art

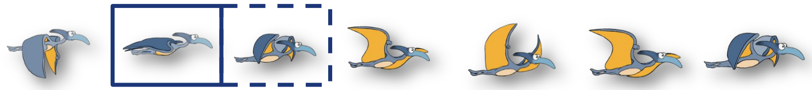


Game Structure



Main classes

```
class AnimatedSprite{
    public:
        AnimatedSprite(SDL_Renderer*& renderer, std::string filepath);
        ~AnimatedSprite();
        void Draw(int x, int y, int w, int h);
        void PlayFrame(int x, int y,int w, int h, int frame);
        void Update();
        void Render(SDL_Renderer*& renderer);
        SDL_Rect getRect() { return m_dst; }

    private:
        SDL_Rect m_src;
        SDL_Rect m_dst;
        SDL_Texture* m_texture;
};
```



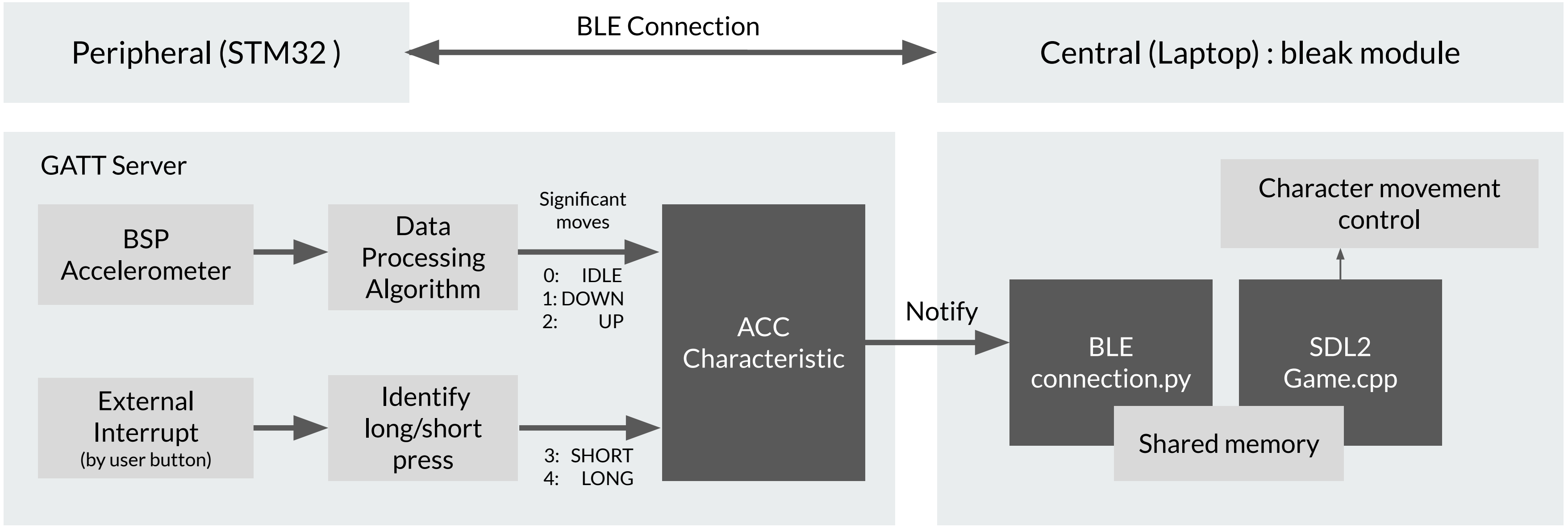
Dealing with objects

- Scrolling objects
    - Different scrolling offsets for relative speeds
- Random object generation
    - Prevent empty map
- Collision detection
    - Bounding box
- Character lane movement
    - Movement constraints and boundaries

Proposed Techniques

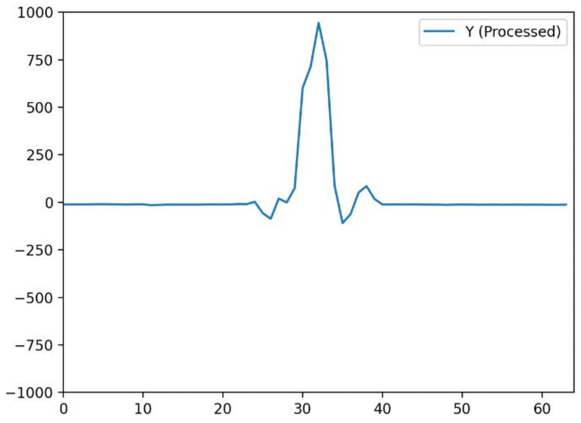
- Peripheral (GATT server) in STM32 IoT node and Central in our computer
  - Increase BSP accelerometer sample rate
- Data Processing:
  - Apply Low Pass Filter on accelerator data (FIR) → **Not ideal after experiment**
  - Weighted Moving Average → **Better for filtering out the noise**
  - Threshold for **avoiding consecutive** out-of-bound data points by windowing
- Game:
  - SDL2 package

Data Flow

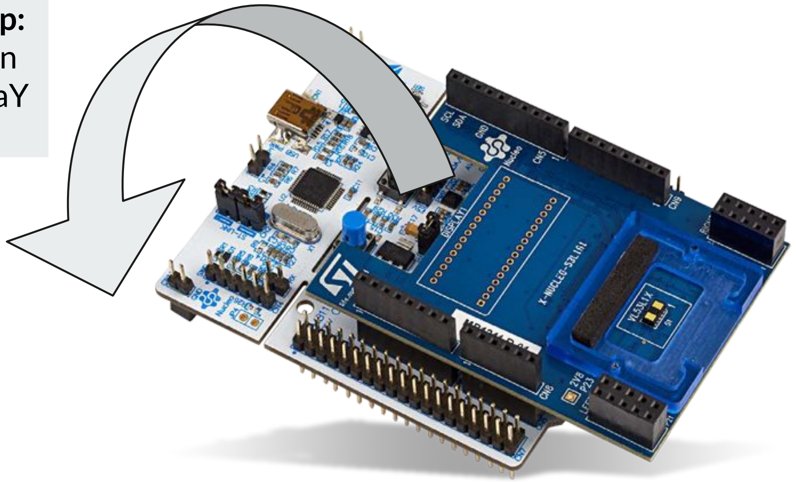


Peripheral (STM32)
BLE Connection
Central (Laptop) : bleak module
GATT Server
BSP Accelerometer
Data Processing Algorithm
Significant moves
0: IDLE
1: DOWN
2: UP
External Interrupt (by user button)
Identify long/short press
3: SHORT
4: LONG
ACC Characteristic
Notify
BLE connection.py
SDL2 Game.cpp
Shared memory
Character movement control

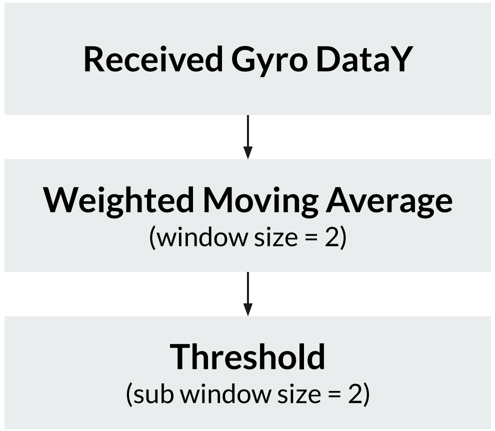
Proposed board movement



Backward Flip: An increase in Accelero DataY



Data Processing Algorithm



for real-time operation, we adjust sampling rate to 6660 Hz.

gives more importance to recent data points by assigning them higher weights compared to older data points within the window, implemented by **deques**

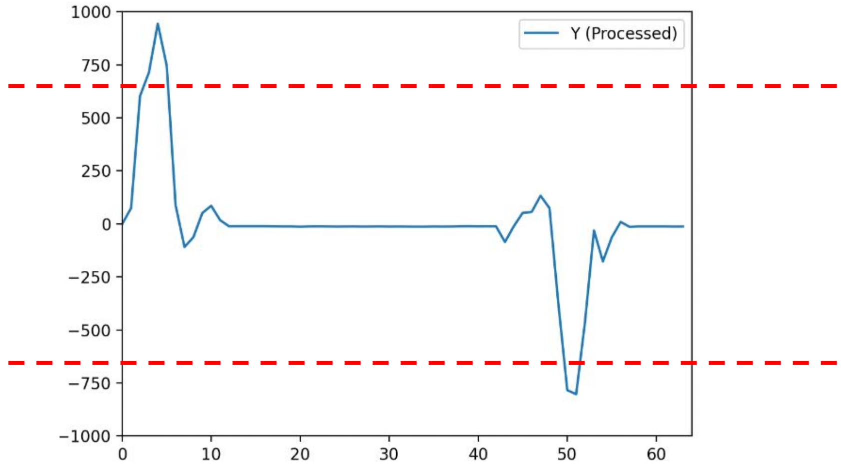
avoid consecutive out-of-bound data points by windowing
Question : Can it still detect successive 'up' or 'down'?

Data Processing Experimental Results

Upper threshold: 950

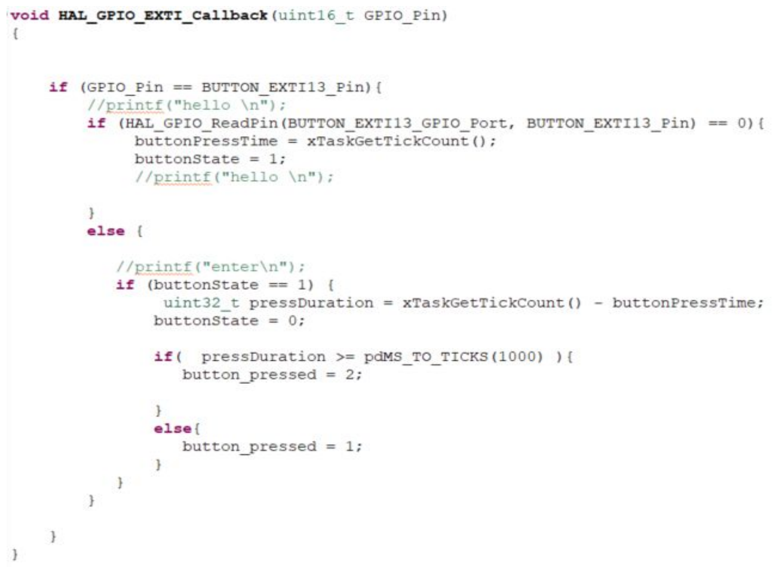
significantly different threshold value why?

Lower threshold: -550



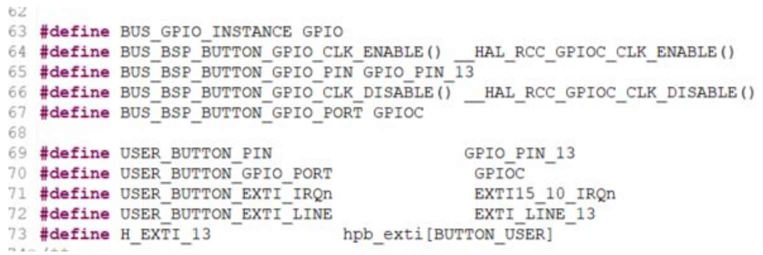
User button settings

- Tap it briefly to shoot fire and destroy obstacles. (denoted as 1)
  - ➢ Send “fire” to the game through BT.
- Press and hold the button for one second before releasing it to recover health. (denoted as 2)
  - ➢ Send “recover” to the game through BT.



```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{


    if (GPIO_Pin == BUTTON_EXTI13_Pin){
        //printf("hello \n");
        if (HAL_GPIO_ReadPin(BUTTON_EXTI13_GPIO_Port, BUTTON_EXTI13_Pin) == 0){
            buttonPressTime = xTaskGetTickCount();
            buttonState = 1;
            //printf("hello \n");

        }
        else {

            //printf("enter\n");
            if (buttonState == 1) {
                uint32_t pressDuration = xTaskGetTickCount() - buttonPressTime;
                buttonState = 0;

                if(  pressDuration >= pdMS_TO_TICKS(1000) ){
                    button_pressed = 2;

                }
                else{
                    button_pressed = 1;
                }
            }
        }

    }
}
```

User button settings



```
#define BUS_GPIO_INSTANCE GPIO
#define BUS_BSP_BUTTON_GPIO_CLK_ENABLE() __HAL_RCC_GPIOC_CLK_ENABLE()
#define BUS_BSP_BUTTON_GPIO_PIN GPIO_PIN_13
#define BUS_BSP_BUTTON_GPIO_CLK_DISABLE() __HAL_RCC_GPIOC_CLK_DISABLE()
#define BUS_BSP_BUTTON_GPIO_PORT GPIOC

#define USER_BUTTON_PIN                     GPIO_PIN_13
#define USER_BUTTON_GPIO_PORT                GPIOC
#define USER_BUTTON_EXTI_IRQn                EXTI15_10_IRQn
#define USER_BUTTON_EXTI_LINE                EXTI_LINE_13
#define H_EXTI_13             hpb_exti[BUTTON_USER]
```

- The GPIO pins defined in the original BLE project might interfere with the operation when using the HAL_GPIO_EXTI_Callback function. Therefore, we commented out all the related definitions of the predefined GPIO pins to avoid conflicts.

DEMO video

Work distribution

- STM32CubeIDE:
  - Data process: 李亦鎧
  - Bluetooth connection: 黃博閩
  - Button press detection: 陳竣瑋
- SDL2 package :
  - Game Logic: 李亦鎧、黃博閩
  - Scrolling Background & Text & Memory : 陳竣瑋
  - Sound Effects & Collision Detection: 李亦鎧
  - Character Animation & Art: 黃博閩、陳竣瑋
- Report :
  - PPT: 李亦鎧、黃博閩
  - Demo Video: 陳竣瑋

References

- https://lazyfoo.net/tutorials/SDL/01_hello_SDL/index.php
- SDL2/Tutorials - SDL Wiki