TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

BÁO CÁO HỆ NHÚNG

Game flappybird trên STM32F429I

Ngành Công nghệ thông tin Việt Nhật

Giảng viên hướng dẫn: Ngô Lam Trung

Nhóm 1:
Nguyễn Xuân An 20204937
Hán Thế Chiến 20205058

Bộ môn: Hệ Nhúng
Trường: Công nghệ thông tin và truyền thông

HÀ NỘI, 7/2023

Mục lục

CHƯƠNG 1. Mục tiêu và nội dung thực hiện trong đồ án

1.1 Mục tiêu

Mục tiêu của đồ án là hiện thực hóa trò chơi kinh điển do người Việt tạo ra là flappybird trên nền tảng vi xử lý STM32F429I

1.2 Nội dung thực hiện trong đồ án

- Đồ họa và giao diện : Xử lý hình ảnh cho nhân vật chim flappybird, hình nền và ống nước
- Xử lý ngõ vào :

  + Đọc các tín hiệu từ nút bấm để điều khiển chuyển động của chim

  + Xử lí thời gian để điều chỉnh tốc độ và khoảng cách giữa các vật thể trong trò chơi
- Quản lí các trạng thái trò chơi như bắt đầu, chơi game, kết thúc.
- Mô phỏng chuyển động của chim Flappy và các ống
- Hiển thị điểm số:

  +Theo dõi và hiển thị điểm số của người chơi trên màn hình.

  + Cập nhật điểm số khi người chơi vượt qua các ống nước.
- Xử lý va chạm:

  +Kiểm tra va chạm giữa chim Flappy và các ống nước hoặc biên của màn hình.

  +Xử lý sự kiện khi va chạm xảy ra, ví dụ: kết thúc trò chơi.
- Radom chiều cao của ống
- Hiệu suất và tối ưu hóa:

  +Đảm bảo rằng trò chơi chạy mượt mà và ổn định trên vi xử lý có tài nguyên hạn chế.

  +Tối ưu hóa mã để sử dụng tài nguyên phần cứng một cách hiệu quả.

CHƯƠNG 2. Thiết kế và xây dựng sản phẩm

Thiết kế game sẽ tương đối giống game flappy kinh điển tuy nhiên sẽ dùng nút bấm để điều khiển chuyển động của chim Flappy

Xây dựng sản phẩm:

- Đồ họa và giao diện:

+ Sử dụng công cụ xử lý hình ảnh chỉnh sửa các hình ảnh chim Flappy, ống, hình nền sao cho phù hợp với kit STM32

- Khai báo các hàm và biến cần sử dụng trong trò chơi

```
class Screen1View : public Screen1ViewBase
{
public:
    Screen1View();
    virtual ~Screen1View() {}
    virtual void setupScreen();
    virtual void tearDownScreen();
    virtual void BirdUp();
    void aaaa();
    void handleTickEvent();
    uint32_t t;
    uint32_t tickCount;
    uint32_t c;
    uint32_t statue;
    int v;
    uint32_t start;
    int tmp;
    int score;
protected:
};
```

- Xử lý ngõ vào:

+ Liên tục polling trạng thái nút bấm trong default task, gửi dữ liệu vào queue

```
/* USER CODE END Header_StartDefaultTask */
void StartDefaultTask(void *argument)
{
  /* USER CODE BEGIN 5 */
  /* Infinite loop */
  for(;;)
  {
    if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0) == GPIO_PIN_SET)
    {
        uint32_t count = osMessageQueueGetCount(Queue1Handle);
        if (count < 2)
        {
            uint8_t x = 'R';
            osMessageQueuePut(Queue1Handle, &x, 0, 200);
        }
    }
    osDelay(10);
  }
  /* USER CODE END 5 */
}
```

- Quản lý trạng thái trò chơi :

Việc quản lý trạng thái trò chơi được sử dụng với 2 biến là Start và statue

```
if(statue!=0 and start!=0){
```

Nếu biến start = 0 thì màn hình sẽ ở vị trí bắt đầu, trò chơi sẽ bắt đầu khi người dùng nhấn nút bấm và chim Flappy sẽ bay lên 1 nhịp

```
if(start==0){
    uint8_t res=0;
        uint32_t count = osMessageQueueGetCount(Queue1Handle);
        if(count>0)
        {
            osMessageQueueGet(Queue1Handle, &res, NULL, osWaitForever);
            if(res == 'R')
                {
                v=2;
                start=1;
                }
        }
}
```

Nếu cả 2 biến statue và start khác 0 thì chương trình sẽ ở trạng thái chơi

statue = 0 khi chim Flappy va chạm với vật thể hoặc rơi xuống và khi đó chương trình sẽ dừng lại

- Mô phỏng chuyển động của chim Flappy, các ống và nền

+Sử dụng biến v để tạo hiệu ứng trọng lực cho chuyển động của chim với biến v thay đổi như khi có gia tốc trọng trường

+ Các ống sẽ chuyển động từ vị trí ban đầu đã được thiết kế lùi về chiều âm của trục X, khi X = -40 (do độ rộng của ống là 40) thì ống sẽ được đặt lại về vị trí 320 để đảm bảo khoảng cách đồng đều giữa các ống

+ Khi bấm nút thì chim sẽ di chuyển lên với vận tốc v = 2

```
c -= statue;
if(c%4==0)v-=1;
tickCount=image10.getY()-v;
image10.setY(tickCount);
image8.setX((image8.getX()-2));
image9.setX((image9.getX()-2));
if(image8.getX() == -40) {
    image8.setX(320);
    image9.setX(320);
}
image11.setX((image11.getX()-2));
image12.setX((image12.getX()-2));
if(image11.getX() == -40) {
        image11.setX(320);
        image12.setX(320);
    }
uint8_t res=0;
uint32_t count = osMessageQueueGetCount(Queue1Handle);
if(count>0)
{
    osMessageQueueGet(Queue1Handle, &res, NULL, osWaitForever);
    if(res == 'R')
        {
        v=2;
        }
}
```

+Hình nền sẽ được hiển thị liên tục mỗi 7 giá trị c để đảm bảo tạo thành chuyển động không quá nhanh cũng không quá chậm

```
switch (c % 49)
{
case 42:
    image1.setVisible(true);
    image2.setVisible(false);
    image3.setVisible(false);
    image4.setVisible(false);
    image5.setVisible(false);
    image6.setVisible(false);
    image7.setVisible(false);
    break;
case 35:
    image1.setVisible(false);
            image2.setVisible(true);
            image3.setVisible(false);
            image4.setVisible(false);
            image5.setVisible(false);
            image6.setVisible(false);
            image7.setVisible(false);
        break;
case 28:
    image1.setVisible(false);
                    image2.setVisible(false);
                    image3.setVisible(true);
                    image4.setVisible(false);
                    image5.setVisible(false);
                    image6.setVisible(false);
                    image7.setVisible(false);
        break;
```

```
case 21:
    image1.setVisible(false);
                            image2.setVisible(false);
                            image3.setVisible(false);
                            image4.setVisible(true);
                            image5.setVisible(false);
                            image6.setVisible(false);
                            image7.setVisible(false);
        break;
case 14:
    image1.setVisible(false);
                            image2.setVisible(false);
                            image3.setVisible(false);
                            image4.setVisible(false);
                            image5.setVisible(true);
                            image6.setVisible(false);
                            image7.setVisible(false);
        break;
case 7:
        image1.setVisible(false);
                                image2.setVisible(false);
                                image3.setVisible(false);
                                image4.setVisible(false);
                                image5.setVisible(false);
                                image6.setVisible(true);
                                image7.setVisible(false);
            break;
case 0:
            image1.setVisible(false);
                                    image2.setVisible(false);
                                    image3.setVisible(false);
                                    image4.setVisible(false);
                                    image5.setVisible(false);
                                    image6.setVisible(false);
                                    image7.setVisible(true);
                break;
}
```

- Xử lý va chạm

```
uint32_t bird[4];
uint32_t pipe[4][4];
bird[0]=image10.getX();
bird[1]=image10.getY();
bird[2]=image10.getWidth();
bird[3]=image10.getHeight();
if(image10.getY()>=288) statue=0;
pipe[1][1]=image8.getY();
pipe[0][0]=image9.getX();
pipe[0][1]=image9.getY();
pipe[0][2]=image9.getWidth();
pipe[0][3]=image9.getHeight();
if (bird[0]+bird[2]>=pipe[0][0] and bird[0]<=pipe[0][0]+pipe[0][2] and (bird[1]<=pipe[0][1]+pipe[0][3] or bird[1]+bird[3] >= pipe[1][1])) statue = 0;
pipe[3][1]=image11.getY();
pipe[2][0]=image12.getX();
pipe[2][1]=image12.getY();
pipe[2][2]=image12.getWidth();
pipe[2][3]=image12.getHeight();
if (bird[0]+bird[2]>=pipe[2][0] and bird[0]<=pipe[2][0]+pipe[2][2] and (bird[1]<=pipe[2][1]+pipe[2][3] or bird[1]+bird[3] >= pipe[3][1])) statue = 0;
```

Lấy ra vị trí hiện tại của chim Flappy cũng như vị trí của các ống nước

Sau đó xử lý điều kiện va chạm của chim Flappy với ống nước, chim Flappy với viền dưới của màn hình:

+Va chạm với viền dưới màn hình khi cạnh dưới của chim Flappy chạm vào màn hình, tức là tọa độ cạnh dưới sẽ bằng 320 tương ứng với cạnh dưới của màn hình

+ Va chạm với ống nước khi cạnh trên hoặc cạnh dưới hoặc cạnh phải va chạm với ống nước, tức là tọa độ của điểm va chạm sẽ trùng với tọa độ của điểm trên cạnh của ống nước

- Radom chiều cao của ống

Việc radom chiều cao của ống rất tốn tài nguyên, vì thế chúng em đã tận dụng vị trí hiện tại của chim để random 3 trạng thái di chuyển của ống với 0 là 2 ống di chuyển lên trên, 1 là 2 ống giữ nguyên, 2 là 2 ống di chuyển xuống dưới(tất nhiên là so với vị trí trước đó) khi ống đi đến 1 mốc nào đó (ví dụ vị trí có X =290)

```
if(pipe[0][0] == 290){
        tmp=bird[1]%3*50-50;
        image8.setY(205+tmp);
        image9.setY(-165+tmp);
    }
if(pipe[2][0] == 290){
            tmp=bird[1]%3*50-50;
            image11.setY(205+tmp);
            image12.setY(-165+tmp);
    }
```

- Hiển thị kết quả :

Mỗi lần qua 1 ống kết quả sẽ được tính bằng cách vị trí cạnh phải của ống trùng với cạnh trái của chim Flappy, do mỗi lần đi qua chỉ có 1 lần bằng nhau. Sử dụng TextArea với giá trị hiển thị là <value> để hiển thị điểm số qua biến Score:

```
if((bird[0]==pipe[0][0]+pipe[0][2] or bird[0]==pipe[2][0]+pipe[2][2]) and statue == 1) score ++;
Unicode::snprintf(textArea1Buffer, TEXTAREA1_SIZE, "%d",score);
invalidate();
```

Phân chia công việc

Họ tên - MSSV	Công việc
Hán Thế Chiến 20205058	Lên ý tưởng, Xử lí giao diện game, Lập trình
Nguyễn Xuân An 20204937	Lập trình, Kiểm tra chương tình, Làm báo cáo