ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO BÀI TẬP LỚN

BATTLESHIP
(Game bắn tàu bay)

Giảng viên hướng dẫn:
Th.S Nguyễn Đức Tiến

Sinh viên thực hiện:
Nguyễn Văn Nam *20200421*
Trần Anh Quốc *20200514*

Mở đầu

Trò chơi Battle Ship là một game bắn tàu bay cổ điển, trong đó người chơi điều khiển một con tàu vũ trụ và phải bảo vệ Trái đất khỏi làn sóng xâm lược của người ngoài hành tinh. Nó có thể được triển khai trong một hệ thống nhúng sử dụng bộ vi điều khiển STM32F429ZI dựa trên kiến trúc ARM. Báo cáo trình bày về các thành phần trong việc tạo ra một trò chơi tương tác và hấp dẫn về mặt hình ảnh. Các tính năng của trò chơi bao gồm chuyển động của người chơi và kẻ thù, phát hiện va chạm, và vượt qua các điều kiện của trò chơi. Những thách thức như tài nguyên hạn chế đã được giải quyết thông qua tối ưu hóa mã và sử dụng phần cứng hiệu quả. Việc triển khai này cho thấy tiềm năng của các hệ thống nhúng trong các ứng dụng trò chơi.

Mục lục:

I. GIỚI THIỆU

1. Tổng quan

Xây dựng trò chơi BattleShip yêu cầu hiểu được sự phức tạp của tích hợp phần cứng-phần mềm, đồ họa, xử lý thông tin nhập của người dùng và hệ thống thời gian thực. Nó cung cấp trải nghiệm thực tế trong việc thiết kế và triển khai một hệ thống nhúng phức tạp, cho phép chúng ta khám phá sự giao thoa giữa chơi game, điện tử và lập trình. Ngoài ra, việc tạo thành công một trò chơi hấp dẫn và đầy thử thách như BattleShip thể hiện khả năng sáng tạo, khả năng giải quyết vấn đề và chuyên môn kỹ thuật của nhóm.

2. Mô tả chi tiết

Luật chơi:

Trong game, người chơi sẽ điều khiển một con tàu vũ trụ và tiêu diệt những kẻ xâm lược mang tên Enemies. Người chơi có thể di chuyển phi thuyền sang trái hoặc phải để tránh đạn từ quân xâm lược. Tàu vũ trụ cũng có thể bắn đạn để tiêu diệt quân xâm lược. Nếu tất cả những kẻ xâm lược bị tiêu diệt, người chơi sẽ thắng.

Trong khi người chơi đang chơi, những kẻ xâm lược sẽ di chuyển qua lại trên màn hình và bắn đạn để tấn công tàu vũ trụ của người chơi. Nếu tàu vũ trụ của người chơi bị bắn hạ hoặc va chạm với kẻ xâm lược, người chơi sẽ mất điểm sinh lực (health point - HP). Nếu người chơi mất hết HP, trò chơi kết thúc và người chơi thua cuộc.

Trong dự án này, chúng tôi sử dụng các kỹ thuật lập trình như cấu trúc dữ liệu, mảng, biến toàn cục, hàm và lệnh điều khiển.

Thiết kế:

- Cài đặt và khởi tạo: Khai báo các thư viện và biến toàn cục cần thiết. Chúng tôi khởi tạo cài đặt cho màn hình LCD
- Cấu trúc: Định nghĩa cấu trúc dữ liệu cho các đối tượng game như Enemies, PlayerShip, Item, bullet_enemy, ...
- Logic trò chơi: Kiểm soát tất cả các khía cạnh của trò chơi. Cụ thể, kiểm soát quân xâm lược, tàu boss, tàu của người chơi chơi và đạn. Kiểm tra va chạm giữa các đối tượng và cập nhật HP cũng như trạng thái của các đối tượng.

- Màn hình bắt đầu: Khi chưa chơi trò chơi, màn hình chào mừng sẽ hiển thị. Trong phần này, màn hình hiển thị nút bắt đầu chơi, yêu cầu người chơi nhấn nút để bắt đầu chơi.
- Màn hình hiển thị: Cập nhật màn hình LCD để hiển thị trạng thái hiện tại của trò chơi. Chúng tôi hiển thị thông tin về các đối tượng, thanh HP, người chơi, tàu kẻ địch và các đối tượng khác như các item hỗ trợ (bắn 3 tia).

Trong trò chơi, có hai loại quân xâm lược: tàu con và trùm cuối. Con tàu đơn lẻ với các loại khác nhau (blue enemy, orange enemy, red enemy) sẽ di chuyển qua lại trên màn hình và bắn đạn để tấn công phi thuyền của người chơi theo mỗi round khác nhau. Người chơi cần tiêu diệt hết các tàu con ở mỗi round trước khi chuyển sang round kế tiếp. Tàu bay của trùm cuối sẽ xuất hiện khi người chơi vượt qua tất cả các round bằng cách tiêu diệt hết các tàu con.

Tàu bay trùm cuối to và mạnh hơn các tàu con, nếu người chơi tiêu diệt được trùm cuối, màn hình chúc mừng chiến sẽ hiện ra cũng với một nút play again để bắt đầu lại trò chơi, nếu người chơi hết HP và bị thua, màn hình game over sẽ hiện ra. Trong game còn có các hiệu ứng đạn, các hiệu ứng cháy nổ và các hiệu ứng items hồi phục và cường hóa bản thân. Đạn được bắn ra từ tàu vũ trụ của người chơi để tiêu diệt những kẻ xâm lược. Khi tàu địch bị phá hủy, vụ nổ sẽ xảy ra và hình ảnh vụ nổ sẽ hiển thị trên màn hình. Nếu người chơi di chuyển tàu vũ trụ của mình qua hình ảnh items, người chơi sẽ nhận được thêm sức mạnh như bắn ba tia hay cộng thêm điểm sinh lực. Ngoài ra, trò chơi có các round khác nhau, mỗi round có độ khó tăng dần. Những kẻ xâm lược sẽ di chuyển nhanh hơn và xuất hiện thường xuyên hơn ở các cấp độ sau.

3. Phân chia công việc

No	Công việc	Thành viên	
		Nguyễn Văn Nam	Trần Anh Quốc
1	Tìm hiểu và chọn đề tài	50%	50%
2	Thiết kế: Nhân vật (Tàu người chơi, Tàu kẻ địch, Trùm cuối, Items tăng cường sức mạnh, Đạn dược, Hiệu ứng nổ)	90%	10%
3	Thiết kế: Hàm điều khiển nhân vật, Khởi tạo, xóa nhân vật, kẻ địch trên màn hình, Hiệu ứng nổ, màn hình bắt đầu, một số di	90%	10%

	chuyển của kẻ địch trong round đánh tàu con.		
4	Thiết kế: Hàm di chuyển của kẻ địch, hiệu ứng di chuyển của các tàu con trong các round đánh tàu con, kẻ địch tấn công, kiểm tra va chạm, tiêu diệt địch.	90%	10%
5	Thiết kế: Round trùm cuối, hoạt ảnh nhân vật, di chuyển, tấn công, hiệu ứng kết liễu, kiểm tra va chạm, màn hình game over, items tăng cường sức mạnh	90%	10%
6	Kiểm thử	50%	50%
7	Tổng hợp báo cáo và trình bày	50%	50%

II. Thiết bị và công nghệ

1. Thiết bị

STM32 là dòng chip 32bit sử dụng công nghệ lõi ARM Cortex mạnh mẽ, hiệu năng tốt nhưng vẫn giữ được giá thành rẻ. Phù hợp với đa số các công ty hiện nay

Các lý do nên chọn STM32 đó là:

- Tốc độ xử lý cao, ngoại vi hỗ trợ rất nhiều, dòng chip phân khúc thấp là STM32F0x cũng có thể hoạt động lên tới 48Mhz, 64kB Flash, 16kB RAM, 8 bộ Timer 16 bit, 1 bộ Timer 32 bit, 10 bộ ADC 12 bit, 8 bộ USART, 2 bộ SPI, 2 bộ I2C.
- Giá thành rẻ nhưng hiệu quả đem lại lớn.
- Học lập trình STM32 rất dễ dàng do cộng đồng hỗ trợ nhiều.
- Dễ xin việc do các công ty vừa và nhỏ sử dụng STM32 trong các ứng dụng rất nhiều
- Công cụ lập trình đều Free và đầy đủ tài liệu hỗ trợ
- Nếu đem STM32 ra so sánh với các dòng chip khác sẽ vẫn có nhiều khuyết điểm. Thế nhưng mặt bằng chung STM32 vẫn là lựa chọn tối ưu khi học lập trình.

2. Công nghệ

STM32CubeIDE

Bộ công cụ của STM32 dựa trên 4 bước cơ bản cấu thành 1 dự án nhúng đó là:

- Configuration với STM32 CubeMX: Bước cấu hình, chọn cho dự án dòng chip đáp ứng được nhu cầu, sau đó cài đặt ban đầu cho các ngoại vi của chip đó. Được trang bị Graphic UI khiến lập trình viên có thể config các ngoại vi hoạt động mà không cần sử dụng code, tăng hiệu xuất làm việc
- Developement với STM32 CubeIDE: Bước phát triển hay việt nam hay gọi là lập trình. Ccũng giống như những IDE (Integrated Development Environment) khác như Keil C, True Studio, Eclipse… STM32 CubeIDE cung cấp cho chúng ta một môi trường lập trình tích hợp, đầy đủ các công cụ để lập trình và phát triển phần mềm nhúng
- Programing với STM32 CubeProg: Nạp chương trình thông qua nhiều chuẩn như SW, UART, OTA, USB. Thông thường chúng ta sẽ không sử dụng phần mềm này vì IDE đã tích hợp tính năng nạp SW (Serial Wire) sử dụng ST – Link và J – link. Ngoài ra còn có thể đọc, ghi và xác minh các thiết bị và bộ nhớ ngoài thông qua nhiều chuẩn giao thức truyền thông có sẵn như JTAG, SWD, UART, USB DFU, I2C, SPI, CAN
- Monitor với STM32 CubeMonitor: Cũng như tên gọi STM32 CubeMonitor giúp chúng ta giám sát ứng dụng đang chạy với nhiều màn hình, hiển thị nhiều thành phần khác nhau trong mạch nhúng

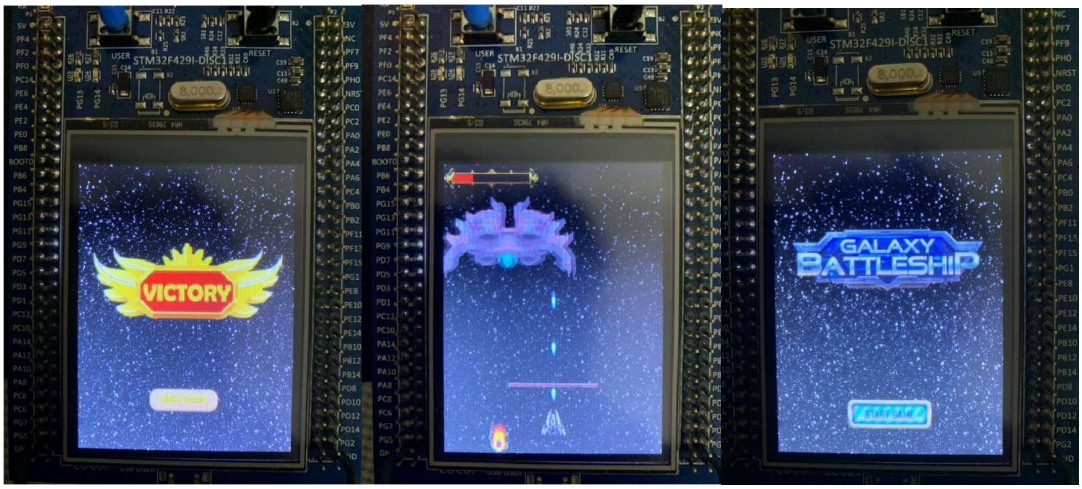
Các gói phần mềm hỗ trợ lập trình nhúng (Package) của hệ sinh thái STM32 Cube

- STM32Cube MCU và MPU Packages: Hỗ trợ xây dựng các driver dựa trên thư viện HAL (Hardware Abstraction Layer) cho từng dòng chip. Hỗ trợ các Middle ware được config tương thích với dòng chip đó. Các gói package này có thể được cài tự động hoặc cài bằng tay bởi các phần mềm bên trên
- STM32Cube Expansion Packages, dành cho các giải pháp ứng dụng. Đây là các package mở rộng của STM32Cube MCU nhằm cung cấp thêm các thành phần phần mềm nhúng, các package này có thể được thiết kế bởi hãng ST hoặc các đối tác của họ để tạo ra thêm các phần mềm nhúng cho

dòng vi điều khiển STM32 để gia tăng sự tiện nghi khi lựa chọn sử dụng dòng vi điều điều STM32.

III. Triển khai dự án

1. Hình ảnh tổng quan



2. Thiết lập đồ họa

Sử dụng TouchGFX.

3. Một số chức năng chính

a. Tàu con (Kẻ xâm lược)

Chúng tôi sử dụng hàm "**moveStraight**" cho phép tàu của địch di chuyển theo hướng chỉ định.

```
void moveStraight(touchgfx::Image &enemy, int16_t stepSize, int16_t
tickCount, int16_t x_A, int16_t y_A, int16_t x_B, int16_t y_B)
{
    int16_t vector_a = x_A - x_B;
    int16_t vector_b = y_A - y_B;
    enemy.setX(x_A - tickCount * vector_a / 50);
    enemy.setY(y_A - tickCount * vector_b / 50);
}
```

Cụ thể như sau:

- Hàm nhận vào tham số là đối tượng cần di chuyển **'enemy'**, biến đếm thời điểm **'tickCount'**, tọa độ x, y hiện tại **'x_A'**, **'y_A'**, tọa độ x, y điểm đến **'x_B'**, **'y_B'**
    - 'tickCount' là biến lưu thời điểm hiện tại của vòng chơi,vòng lặp
- Tính tọa độ vector di chuyển bằng các lấy tọa độ điểm hiện tại trừ tọa độ điểm đến

    - vector_a = x_A - x_B;
    - vector_b = y_A - y_B;
- Phương trình tọa độ:
    - x = x_A – vector_a * (tickCount/50)
    - y = y_A – vector_b * (tickCount/50)
- Di chuyển tàu địch đến vị trí mới theo theo thời gian dựa vào phương trình trên.

Chúng tôi sử dụng hàm "**enemyAttack**" cho phép tàu của địch tấn công bằng cách bắn đạn theo hướng thẳng xuống.

```
void enemyAttack(
touchgfx::Image &enemy, touchgfx::Image &bullet,
int16_t isMove, int16_t bullet_speed, int16_t x_axis)
{
    int16_t x = enemy.getX();
    int16_t y = enemy.getY();
    if (isMove == 1) {
        bullet.setY(bullet.getY() + bullet_speed);
    }
    if (isMove == 0) {
        bullet.setXY(x + 3, y + 18);
    }
}
```

Cụ thể như sau:

- Hàm nhận vào tham số là đối tượng tàu địch **'enemy'**, đối tượng đạn **'bullet'**, tham số kiểm soát **'isMove'** để kiểm soát liệu tàu có được bắn đạn hay không, tốc độ đạn bay **'bullet_speed'**, tham số **'x_axis'** chỉnh độ lệch tọa độ đạn so với tàu.
- Tọa độ tàu địch được lưu trong các biến x, y.
- Nếu 'isMove' bằng 1 (tức là tàu đã bắn đạn ra và đạn đang trên đường bay) thì cập nhật vị trí đạn bay thẳng xuống theo chiều dọc.
- Nếu 'isMove' bằng 0 (tức là đạn đã bay hết đường) thì cho đặt vị trí đạn trước mũi tàu địch (tàu địch bắn đạn) .

b. Người chơi

Chúng tôi tạo sử dụng hàm "**control**" để xử lí điều khiển từ người chơi và cho phép tàu của người chơi di chuyển sang trái hay sang phải.

```
void control(
touchgfx::Image &player,
touchgfx::BoxWithBorderButtonStyle<touchgfx::RepeatButtonTrigger>rightButton,
touchgfx::BoxWithBorderButtonStyle<touchgfx::RepeatButtonTrigger>leftButton)
{
    int16_t x = player.getX();
    if(rightButton.getPressed()==1){
        x+=3;
        if(x>240){
```

```
            x=0;
        }
        player.setX(x);
        }
    if(leftButton.getPressed()==1){
        x-=3;
        if(x<0){
            x=240;
        }
        player.setX(x);
    }
}
```

Cụ thể như sau:

- Tọa độ hiện tại của tàu được lưu vào biến 'x' .
- Nếu **'rightButton'** được kích hoạt (rightButton.getPressed() trả về giá trị 1) thì tăng giá trị biến 'x' (vị trí bên phải so với vị trí hiện tại), nếu giá trị này vượt qua 240 (tàu đi quá giới hạn bên phải của màn hình hiển thị) thì đặt lại giá trị 'x' bằng 0 (sát phía trái của màn hình), sau đó cập nhật vị trí mới của tàu bằng giá trị biến 'x' (player.setX(x)).
- Tương tự với khi nút **'leftButton'** được kích hoạt thì tàu di chuyển về phía trái màn hình và quay trở lại từ phía phải màn hình nếu đi quá giới hạn phía trái màn hình.

c. Kiểm tra đụng độ (Collision Checker)

Chúng tôi tạo hàm 'checkCollision' để kiểm tra xem liệu 2 đối tượng trong trò chơi có đang va chạm với nhau hay không, các va chạm có thể là giữa đạn của tàu địch với tàu của người chơi, giữa đạn của người chơi với tàu của địch, giữa vật phẩm và tàu của người chơi,...

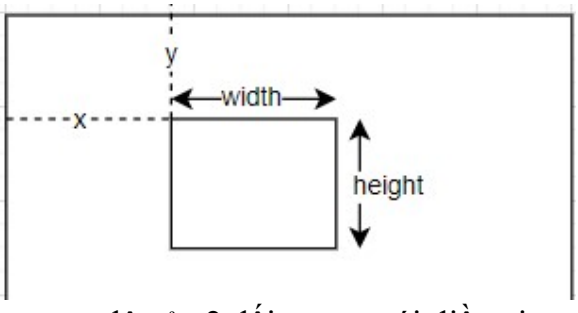
```
bool checkCollision(
const touchgfx::Image &image1, const touchgfx::Image &image2)
{
    touchgfx::Rect image1Rect = image1.getRect();
    int16_t x1 = image1Rect.x;
    int16_t y1 = image1Rect.y;
    int16_t width1 = image1Rect.width;
    int16_t height1 = image1Rect.height;
    touchgfx::Rect image2Rect = image2.getRect();
    int16_t x2 = image2Rect.x;
    int16_t y2 = image2Rect.y;
    int16_t width2 = image2Rect.width;
    int16_t height2 = image2Rect.height;

    if (x1 < (x2 + width2) && (x1 + width1) > x2
     && y1 < (y2 + height2) && (y1 + height1) > y2)
    {
        return true;
    }
```

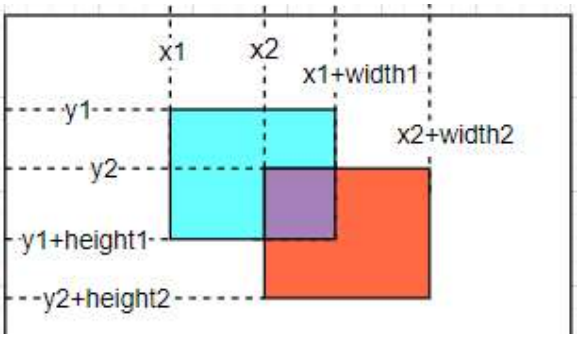
```
    return false;
}
```

Cụ thể như sau: Here is a description of this function:

- Hàm nhận vào 2 đối tượng cần kiểm tra va chạm
- Tọa độ của vật thể là hình chữ nhật được đặc trưng bởi 4 thuộc tính (x, y, width, height). Trong đó,
    - x là tọa độ điểm bắt đầu theo chiều ngang
    - y là tọa độ điểm bắt đầu theo chiều dọc
    - width là chiều rộng theo chiều ngang của đối tượng
    - height là chiều cao theo chiều đọc của đối tượng



- Hàm sẽ kiểm tra tọa độ của 2 đối tượng với điều giao nhau của 2 hình chữ nhật trong mặt phẳng tọa độ. Nếu có va chạm giữa 2 đối tượng, hàm trả về kết quả **'true'**, nếu không trả về **'false'.**



Chúng tôi sử dụng hàm **"remove"** và hàm **"explosion"** để loại bỏ tàu địch và gây hiệu ứng tàu nổ khi đạn của người chơi trúng tàu địch.

```
void remove(touchgfx::Image &enemy, touchgfx::Image &bullet) {
    enemy.setXY(400, 300);
    bullet.setXY(400, 300);
    bullet.setVisible(false);
    enemy.setVisible(false);
}
void explosion(touchgfx::Image &enemy, touchgfx::AnimatedImage explore)
{
    explore.setXY(enemy.getX(), enemy.getY());
    explore.setVisible(true);
}
```

Cụ thể như sau:

- Hàm **"remove"** nhận vào tham số là đối tượng tàu địch **'enemy'** và đối tượng đạn của người chơi **'bullet'.**
- Hàm **"explosion"** nhận vào tham số là đối tượng tàu địch **'enemy'** và hiệu ứng nổ **'explore'.**
- Khi có tàu địch va chạm với đạn của người chơi hàm **"remove"** và hàm **"explosion"** sẽ được gọi để loại 2 đối tượng này khỏi màn hình trò chơi và gây hiệu ứng tàu nổ tại vị trí tàu địch bị bắn trúng.
- Việc loại bỏ được thực hiện bằng cách đặt tọa độ tàu địch và đạn ra khỏi phạm vi màn hình hiển thị.

d. Trò chơi

Hàm "**handleTickEvent"** để cập nhật trò chơi trên màn hình hiển trị

Cụ thể như sau:

1. Kiểm tra lượng máu của người chơi

- Nếu lượng máu của người chơi nhỏ hơn hoặc bằng 0, chuyển đến màn hình kết thúc trò chơi.
- Nếu không thì trò chơi vẫn tiếp tục.

2. Tự động rớt vật phẩm nâng cấp cho người chơi

```
if (tickCount % 100 == 0) {
    isDrop = 1;
}
if (isDrop == 1) {
    x3.setY(x3.getY() + 5);
}
if (x3.getY() > 320) {
    isDrop = 0;
    x3.setY(-20);
}
if (checkCollision(x3, playerShip)) {
    isItem = 1;
    x3.setY(-20);
    isDrop = 0;
}
if (isItem == 1) {
    countItem += 1;
    if (isMove[10 + tickCount / 10 % 10] == 0) {
        isMove[10 + tickCount / 10 % 10] = 1;
        isMove[20 + tickCount / 10 % 10] = 1;
        bullet_list[10 + tickCount / 10 % 10].setXY(x - 3, y - 15);
        bullet_list[20 + tickCount / 10 % 10].setXY(x + 18, y - 15);
    }
    for (int16_t i = 10; i < 30; i++) {
        bullet_list[i].setY(bullet_list[i].getY() - 5);
        if (bullet_list[i].getY() <= -23) {
            isMove[i] = 0;
```

```
        }
    }
    if (countItem >= 100) {
        isItem = 0;
        countItem = 0;
        for (int16_t i = 10; i < 30; i++) {
            bullet_list[i].setY(-50);
        }
    }
}
```

- Sau một khoảng thời gian (được đánh dấu bởi biến đến thời gian 'tickCount'), sinh ra vật phẩm nâng cấp cho người chơi.
- Vật phẩm được sinh ra sẽ rớt xuống theo chiều dọc và biến mất khi va chạm với tàu của người chơi.
- Khi người chơi chạm được vật phẩm, tàu nhận được nâng cấp khả năng bắn 3 đường đạn trong 1 khoảng thời gian.

3. Tàu của người chơi bắn đạn tự động

```
    //Move the bullet of player
    if (isMove[tickCount / 10 % 10] == 0) {
        isMove[tickCount / 10 % 10] = 1;
        bullet_list[tickCount / 10 % 10].setXY(x + 8, y - 15);
    }
    for (int16_t i = 0; i < 10; i++) {
        bullet_list[i].setY(bullet_list[i].getY() - 5);
        if (bullet_list[i].getY() <= -23) {
            isMove[i] = 0;
        }
    }
```

- Luồng đạn của tàu được lưu trong danh sách **'bullet_list'**, tối đa 10 viên đạn.
- Danh sách 'isMove' lưu trạng thái của đạn, 1 là đang trên đường bay, 0 là sẵn sàng để bắn.
- Nếu đạn sẵn sàng để bắn thì tàu bắn ra đạn từ mũi tàu và di chuyển lên trên theo chiều dọc,lúc này đạn ở trạng thái đang trên đường bay.
- Khi đạn di chuyển ra khỏi phạm vi hiển thị của màn hình, đạn trở về trạng thái sẵn sàng để bắn.

4. Tàu của địch tự động bắn đạn

```
for (int16_t i = 0; i < 9; i++) {
        if (isActive[i]) {
            if (bullet_enemy[i].getY() >= 320) {
                enemyAttack(enemy_list[i], bullet_enemy[i], 0, 5, 3);
```

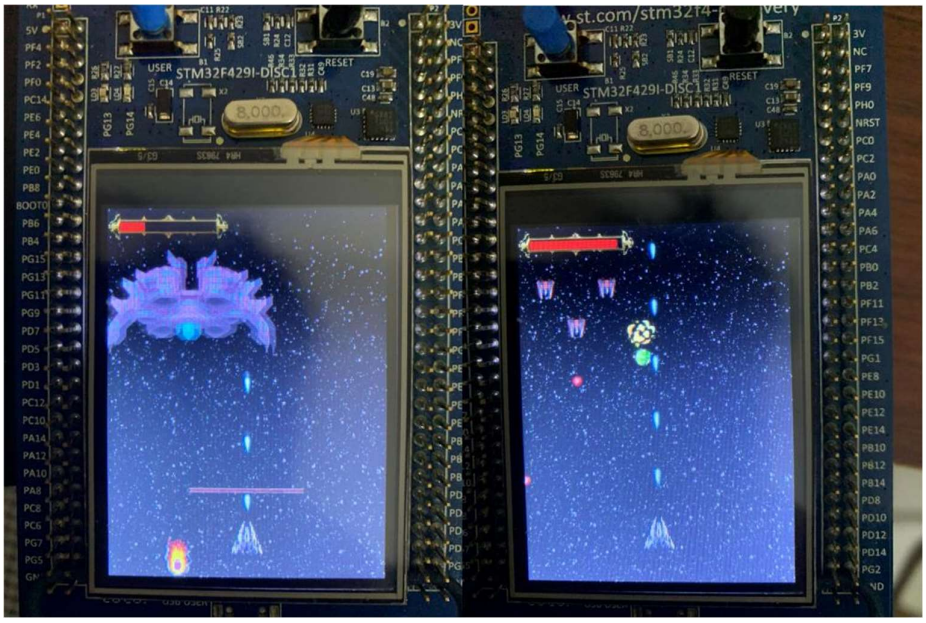
```
            } else {
                enemyAttack(enemy_list[i], bullet_enemy[i], 1, 5, 3);
            }
            if (checkCollision(bullet_enemy[i], playerShip)) {
                playerHP = playerHP - 5;
                hpplayer.setPosition(10, 11, playerHP, 9);
                bullet_enemy[i].setY(320);
            }
        }
    }
```

- Danh sách ‘enemy_list’ lưu các đối tượng tàu địch, ‘bullet_enemy’ lưu các đối tượng đạn của tàu địch, ‘isActive’ lưu trạng thái đang bay hay đã bị hủy của tàu địch.
- Kiểm tra trạng thái ‘active’ của tàu địch, nếu tàu địch còn bay thì cho tàu địch bắn đạn, xuống theo chiều dọc.
- Khi đạn đi quá phạm vi hiển thị của màn hình, thì tàu sẵn sàng bắn đạn tiếp theo.
- Khi tàu người chơi va vào đạn thì người chơi bị trừ máu và đạn bị huỷ.

IV. Demo

1. Menu

2. Màn hình cho người chơi



3. End Game


STM32F429I-DISC1
RESET
USER
8.000
VICTORY

V. Kết luận

Tóm lại, thực hiện dự án xây dựng trò chơi BattleShip như một phần của khóa học hệ thống nhúng là một nỗ lực thú vị và bổ ích. Nó cho phép chúng em kết hợp niềm đam mê chơi game với các kỹ năng kỹ thuật có được thông qua khóa học. Bằng cách phát triển trò chơi BattleShip, chúng em có được kinh nghiệm thực hành về tích hợp phần cứng-phần mềm, thiết kế đồ họa, và các hệ thống thời gian thực. Hơn nữa, việc tạo ra một trò chơi hấp dẫn và đầy thử thách như BattleShip thể hiện khả năng của bạn trong việc áp dụng kiến thức hệ thống nhúng vào các ứng dụng thực tế và hấp dẫn. Nhìn chung, xây dựng trò chơi BattleShip trong một khóa học về hệ thống nhúng là một cơ hội tuyệt vời để nâng cao kỹ năng của chúng em, khám phá sự giao thoa giữa trò chơi và công nghệ, đồng thời chuẩn bị cho những nỗ lực trong tương lai trong lĩnh vực phát triển trò chơi hoặc kỹ thuật hệ thống nhúng.