**Đại học Bách Khoa Hà Nội**

**Trường công nghệ thông tin và truyền thông**

Logo

Description automatically generated with medium confidence

**Báo cáo đồ án cuối kì**

**Đề tài: Trò chơi Madness Road trên kit STM32F429I-DISC1**

**Môn học: Hệ nhúng**

Nhóm 10:

|  |  |
| --- | --- |
| Họ tên | Mã số sinh viên |
| Trịnh Quang Hùng | 20190054 |
| Lê Quốc Hưng | 20191881 |
| Bùi Ngọc Tân | 20190067 |
| Nghiêm Việt Thắng | 20190088 |

Giảng viên hướng dẫn: TS. Đỗ Công Thuần

Hà Nội, ngày 13 tháng 3 năm 2023

Mục lục

[**I.** **Phân công công việc** 3](#_Toc129585857)

[**II.** **Mô tả đề tài** 3](#_Toc129585858)

[1. Giới thiệu trò chơi 3](#_Toc129585859)

[2. Công nghệ sử dụng 4](#_Toc129585860)

[**III.** **Thiết kế** 5](#_Toc129585861)

[1. Thiết kế sơ đồ mạch 5](#_Toc129585862)

[2. Thiết kế giao diện 5](#_Toc129585863)

[**IV.** **Lập trình** 7](#_Toc129585864)

[1. Xử lý di chuyển của các đối tượng 7](#_Toc129585865)

[2. Xử lý tốc độ trong trò chơi 7](#_Toc129585866)

[4. Cách xuất hiện của các đối tượng 8](#_Toc129585867)

[5. Xử lý các thông số khác 9](#_Toc129585868)

[**V.** **Kết luận** 9](#_Toc129585869)

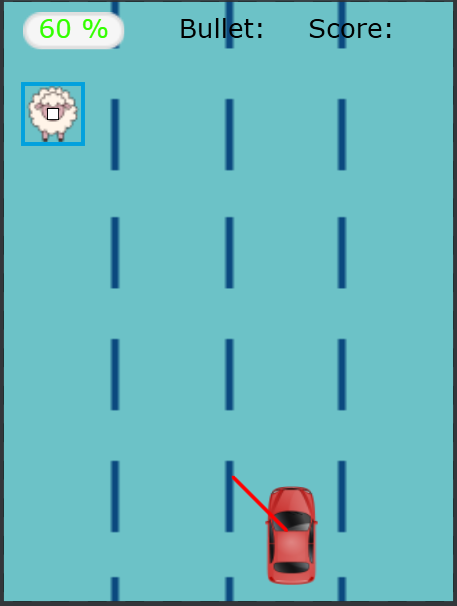
1. **Phân công công việc**

|  |  |
| --- | --- |
| **Họ tên thành viên** | **Công việc** |
| Trịnh Quang Hùng | Thiết kế và lập trình game lái xe vượt chướng ngại vật cơ bản và luồng cho các chức năng khác |
| Lê Quốc Hưng | Nghiên cứu và lắp mạch, thêm tương tác di chuyển đặc biệt cho các chướng ngại vật |
| Bùi Ngọc Tân | Thiết kế và lập trình cho UI bên ngoài và chức năng bay cho xe |
| Nghiêm Việt Thắng | Lập trình cho chức năng bắn đạn và độ khó theo cấp độ điểm số |

1. **Mô tả đề tài**
2. Giới thiệu trò chơi

**Madness Road** là trò chơi thể loại đua xe tránh chướng ngại vật tính điểm với đồ họa 2D góc nhìn từ trên xuống.

Người chơi có nhiệm vụ điều khiển xe để né những chướng ngại vật (các con cừu) có trên đường bằng cách điều khiển xe sang trái, phải hoặc cho xe bay lên. Ngoài ra xe còn có thể bắn đạn theo hướng tia màu đỏ trước đầu xe để diệt cừu. Khi người chơi vượt qua hoặc tiêu diệt một con cừu (kể cả bằng cách tông vào nó) sẽ có thêm 1 điểm. Nếu không thể né/bắn chết cừu trước khi xe đụng phải, người chơi sẽ bị mất 1 mạng. Nếu mất hết mạng, trò chơi sẽ kết thúc.



Hình 1: Giao diện màn chơi của Mad Road

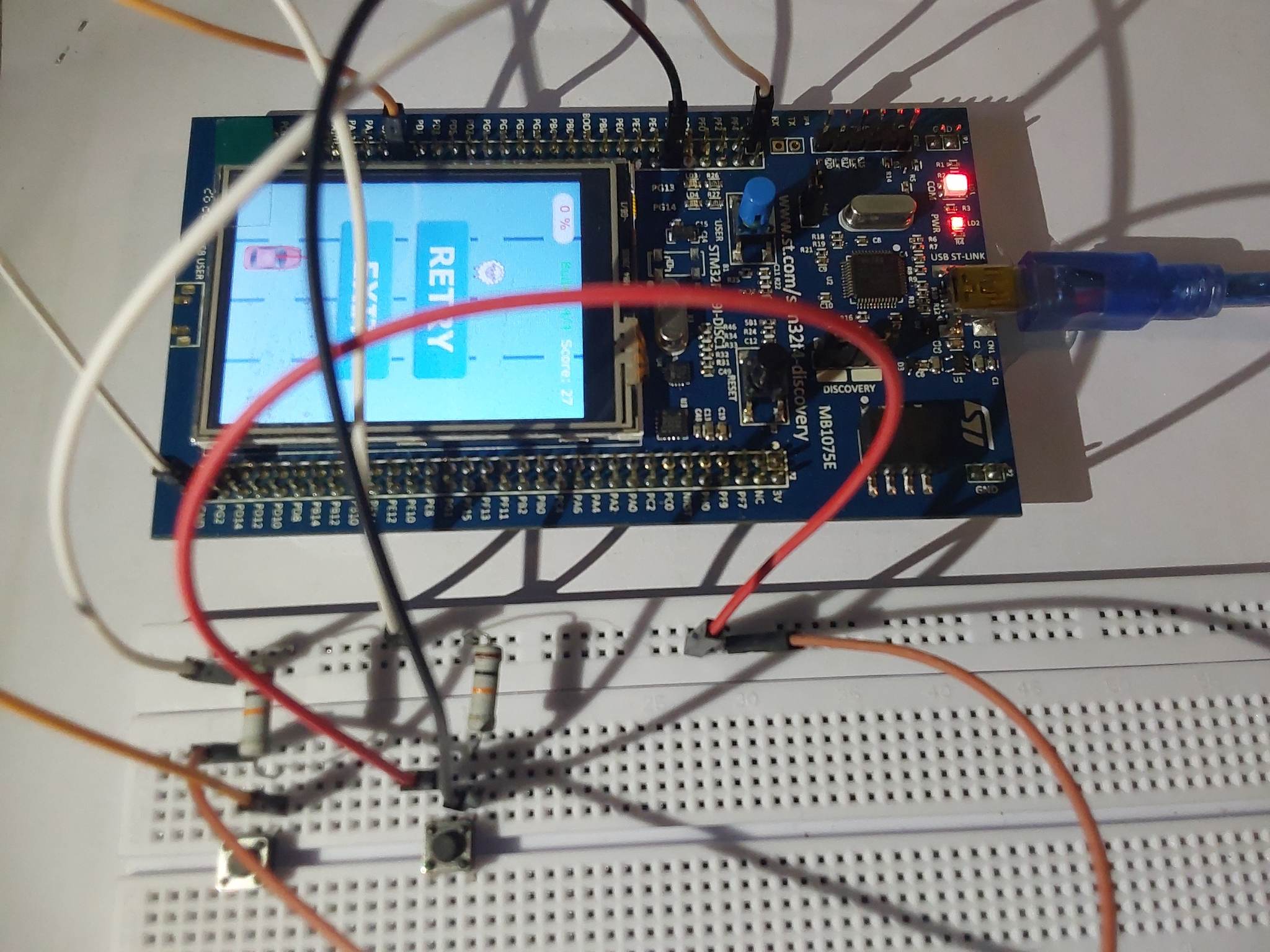
1. Công nghệ sử dụng

TouchGFX: là một framework đồ họa nhúng của STMicroelectronics, được sử dụng để tạo giao diện người dùng trên các thiết bị nhúng. Nó cung cấp các công cụ thiết kế đồ họa và hỗ trợ tối ưu hóa hiệu suất cho ứng dụng đồ họa nhúng. TouchGFX là một giải pháp phù hợp cho các thiết bị nhúng có yêu cầu về đồ họa và giao diện người dùng cao, như các thiết bị y tế, thiết bị điện tử gia dụng, xe hơi, hệ thống điều khiển công nghiệp và nhiều ứng dụng nhúng khác.

STM32CubeIDE: là một môi trường phát triển tích hợp (IDE) cho các dòng vi điều khiển ARM Cortex-M và Cortex-A của STMicroelectronics. Nó cung cấp các công cụ cần thiết cho việc lập trình, gỡ lỗi và triển khai các ứng dụng nhúng trên các thiết bị STM32. STM32CubeIDE bao gồm trình biên dịch GCC, trình gỡ lỗi GDB và các công cụ phát triển phần mềm khác như STM32CubeMX và STM32CubeProfiler. Nó cũng hỗ trợ các tính năng tiên tiến như gỡ lỗi trực tiếp trên thiết bị, tích hợp Git và tạo mô hình thực thi chương trình.

1. **Thiết kế**

## Thiết kế sơ đồ mạch

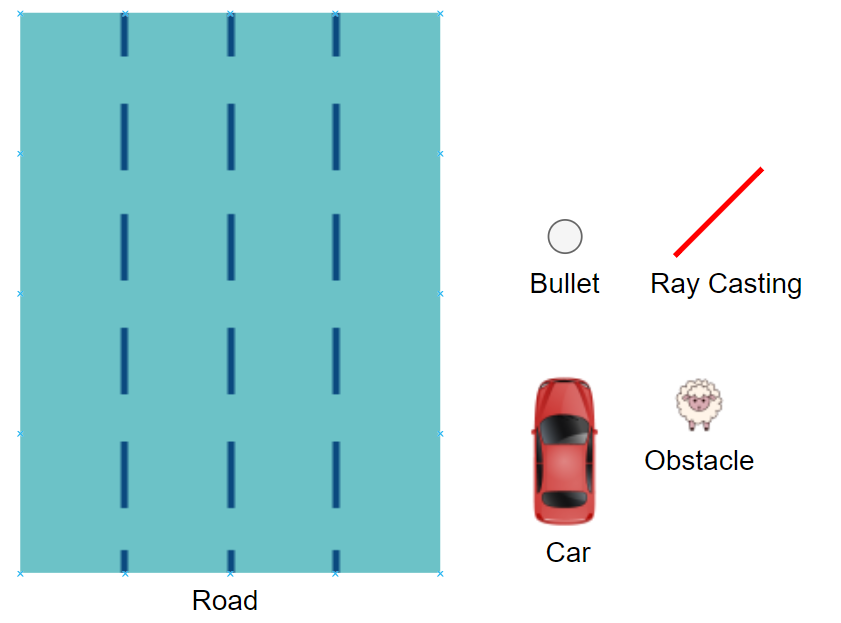


Nhóm sử dụng thiết kế mạch với 3 ngắt ngoài trong với 2 phím bấm được nối với các chân PC11 và PC13 và phím bấm USER\_BUTTON tương ứng với PA0 để tạo các tương tác với trò chơi.

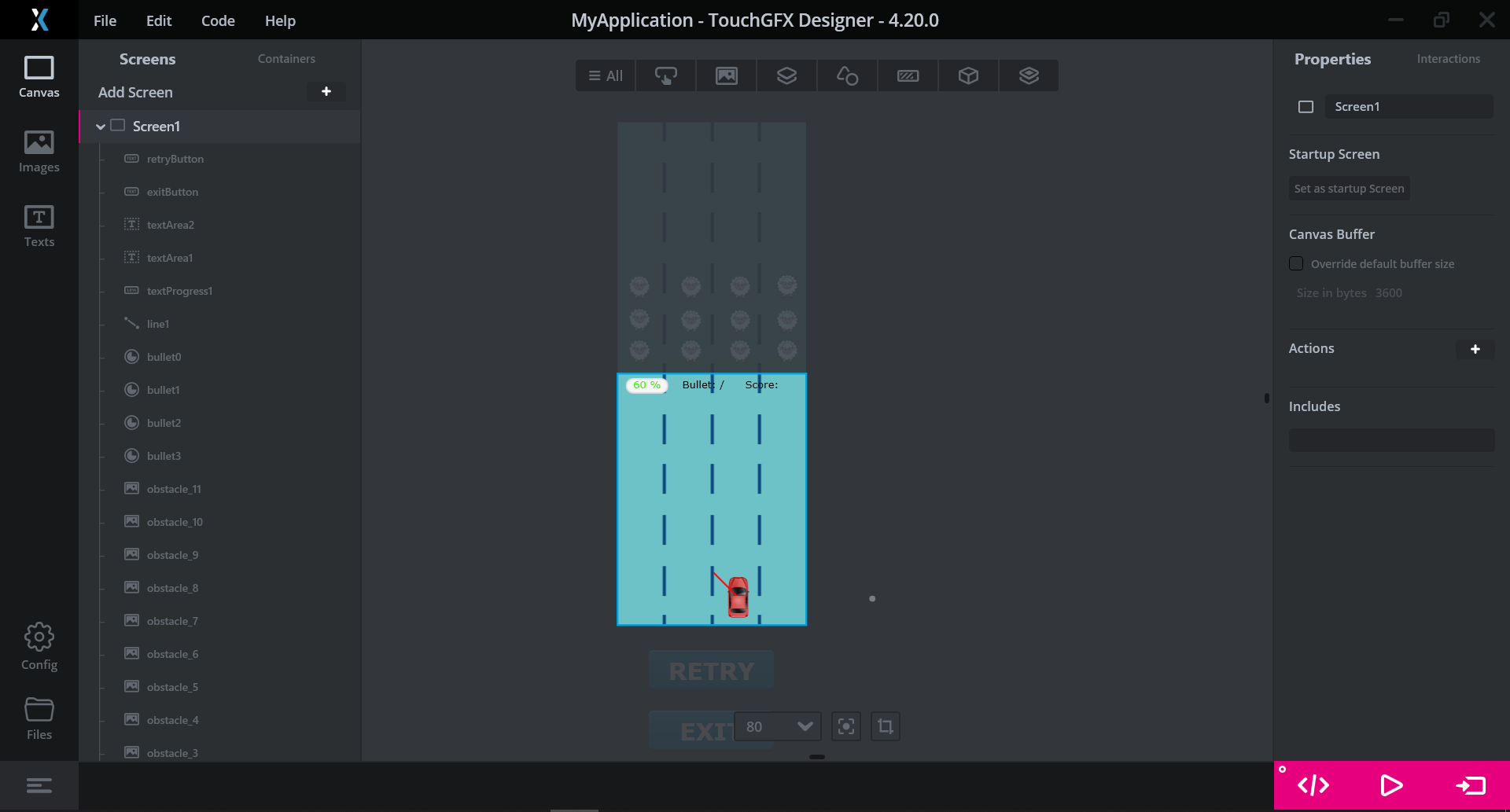
## Thiết kế giao diện

Nhóm thực hiện thiết kế giao diện trong touchGFX trước khi sử dụng STM32CubeIDE để lập trình logic.

Dưới đây là hình ảnh các thành phần trong trò chơi: bao gồm hình ảnh nền phía sau, thiết kế của viên đạn, thước ngắm, xe và chướng ngại vật.



Hình 2: Các thành phần trong trò chơi



Hình 3: Thiết kế trong touchGraphFX

1. **Lập trình**
2. Xử lý di chuyển của các đối tượng

Vật thể người chơi điều khiển bao gồm chiếc xe của mình. Nhưng chiếc xe đã bị cố định tung độ và hoành độ được duy trì trong khoảng từ 0 tới 203. Để tạo cảm giác xe đang chuyển động về phía trước liên tục, chúng ta di chuyển 2 bức ảnh khung cảnh nền phía sau của xe với một tốc độ cố định xuống phía dưới màn hình. Hai bức ảnh nền này khớp liền nhau tạo cảm giác liên tục về mặt đường đi tức phía trên của ảnh này khớp mặt phía dưới của ảnh kia và ngược lại. Ngoài ra, các vật cản (đàn cừu) sẽ được di chuyển xuống phía dưới cùng hướng với ảnh nền với cùng vận tốc vận tốc.

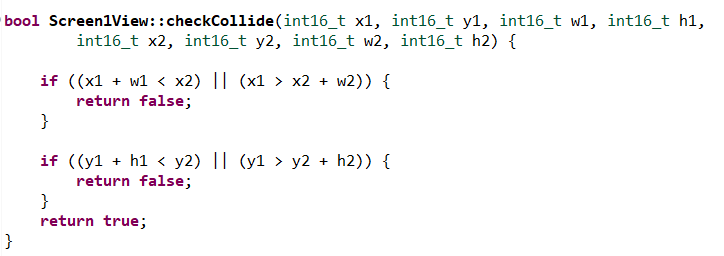
Trong game chỉ có những viên đạn được di chuyển cách riêng biệt với một vector xác định hướng bắn được xác định ngay khi thực hiện thao tác bắn bằng các lấy vector của tia định hướng.

1. Xử lý tốc độ trong trò chơi

Sử dụng 1 biến (speed) làm tốc độ chung cho các đối tượng di chuyển trong màn chơi như khung nền phía sau hay các chướng ngại vật. Để mọi đối tượng di chuyển trong màn chơi chúng sẽ nhận thêm vào một lượng giá trị biến tốc độ nào đó. Ngoài ra, chúng ta sẽ sử dụng biến này để thay đổi độ khó của màn chơi trong quá trình chơi.

1. Xử lý va chạm giữa các đối tượng

Mỗi đối tượng dạng widget được nằm trong một bounding box được xác định bởi tạo độ góc trái bên trên (x, y) và kích thước (w, h) từ đó có thể xác 2 đối tượng va chạm theo công thức sau:

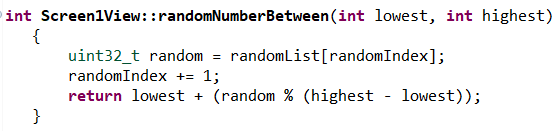


Hình 4: Công thức xác định va chạm giữa các đối tượng

Trong trò chơi, cần phải xác định sự va chạm giữa 2 cặp đối tượng sau đó là giữa viên đạn và cừu cũng như giữa cừu và xe. Do tốc độ của cừu và xe tương đối chậm nên việc xác định va chạm rất dễ dàng nhưng đối với viên đạn tốc độ khi đủ nhanh sẽ vượt qua khả năng check va chạm của hàm trên. Đó là một lỗi logic nhóm đang tìm cách khắc phục.

1. Cách xuất hiện của các đối tượng

Chướng ngại vật trong trò chơi được xuất hiện một cách ngẫu nhiên bằng cách ngẫu nhiên vị trí xuất phát của chúng trong bốn vị trí xuất phát ban đầu. Tham số quyết định thời gian xuất hiện liên tiếp giữa hai vật cản là biến spawnInterval cũng là một tham số nhằm quyết định độ khó của trò chơi.



Hình 5: Công thức của hàm sinh giá trị ngẫu nhiên

Để sinh ngẫu nhiên được một số bất kỳ nhóm sử dụng một mảng gồm 100 các giá trị ngẫu nhiên trong khoảng tư 1 đến 1000 và lấy lần lượt từ đó ra. Do trong quá trình, chạy chương trình số lần lấy phần tử ngẫu nhiên không cố định nên không quá khắt khe, hàm trên cho chúng ta một cách ngẫu nhiên tương đối tốt. Ngoài ra, tham số randomIndex được khởi tạo ngẫu nhiên bằng số lần bấm phím của người dùng trong giao diện Menu nên cũng có thể nói đã đảm bảo được tính ngẫu nhiên tương đối.

Ngoài ra, các hình ảnh của các đối tượng như cừu hoặc đạn đều được khởi tạo một số lượng lớn nhưng chỉ có những đối tượng được triệu hồi ra và trong tầm hiển thị của màn hình mới được vẽ cũng như cập nhật các logic liên quan. Kỹ thuật này có thể giúp chúng ta không phải khởi tạo liên tục các đối tượng xuất hiện nhiều trong quá trình chạy chương trình.

1. Xử lý các thông số khác

Quá trình chơi game được tăng độ khó liên tục bằng cách thay đổi hai tham số về thời gian giữa 2 lần xuất hiện của chướng ngại vật liên tiếp và tham số về tốc độ di chuyển. Do đó, trong quá trình chơi cứ sau 10 điểm số đạt được màn chơi sẽ có sự thay đổi về độ khó nhất định.

Giao diện UI được cập nhật lại mỗi một tick nhằm hiển thị điểm số, lượng máu và số lượng đạn người chơi đang còn lại.

1. **Kết luận**

Quá trình thực hiện đồ án, xây dựng và thiết kế trò chơi Madness Road giúp nhóm được thực hành các nội dung được học trong học phần “Hệ nhúng”, từ đó củng cố lại kiến thức môn học.