Project Instruction

Project: Racing Cars

Mô tả ứng dụng

Ứng dụng cần phải đảm bảo các chức năng cơ bản. Tuy nhiên, bạn có thể thêm chức năng bổ sung vào ứng dụng của bạn, nếu bạn muốn.

Phần 1: Chức năng và yêu cầu cơ bản

---

1. Thiết kế giao diện
    1. Giao diện có màu sắc hài hoà, layout hợp lý và nội dung rõ ràng.
    2. Trong trang web phải có phần mô tả về trò chơi: bao gồm tên trò chơi và các hướng dẫn ngắn gọn về luật chơi.
    3. Để tạo quang cảnh đẹp cho việc đua xe, thì bạn cần vẽ thêm bầu trời màu xanh, đám mây lửng lơ màu trắng, mặt trời thì xanh cao vời vợi còn dưới đất thì cỏ xanh mơn mởn.
    4. Các xe đua phải có màu sắc và kích cỡ đa dạng.

---

2. Yêu cầu chức năng cơ bản

    Xây dựng mini game Racing Cars, bao gồm các chức năng sau

    1. Tạo ra 3 chiếc xe oto
    2. Mỗi oto sẽ có màu sắc, kích thước và tốc độ cũng như vị trí khác nhau
    3. Vị trí xuất hiện của xe là cùng vạch xuất phát và cách nhau một khoảng nhất định (cùng toạ độ x và khác toạ độ y) , và tốc độ xe cũng là ngẫu nhiên
    4. Các xe có khả năng di chuyển liên tục, và nó sẽ dừng lại khi tới đích hoặc chướng ngại vật hoặc xe khác
    5. Xe có khả năng di chuyển theo trục x và cả y, tức là không chỉ đi ngang mà còn lên dốc xuống dốc.
    6. Cuộc đua sẽ được coi là kết thúc khi có xe về đích đầu tiên.
    7. Để cuộc đua thêm lý thú ta cần tạo ra các vụ va chạm: va chạm giữa xe với xe thì cả 2 xe đều bị loại, va chạm giữa xe và chướng ngại vật thì xe cũng bị loại.
    8. Các chướng ngại vật là những tảng đá xuất hiện ngẫu nhiên trên đường đua.

Phần 2. Tổ chức code

Dưới đây là gợi ý tham khảo về cách thức tổ chức code cho trang web. Bạn hoàn toàn có thể tổ chức code theo cách riêng của mình.

- **File car.html**

  Chứa cấu trúc html của trang web

  Có thẻ script cung cấp link tới file thư viện processing.js

  Có thẻ canvas link tới file car.pde chứa kịch bản game

- **File processing.js**

  Thư viện Javascript chứa các hàm có sẵn để vẽ ra các hình, tô màu và text.

  File car.pde sẽ gọi tới thư viện này.

- **File car.pde**

  Chứa kịch bản game để hiển thị trên vùng canvas của file car.html

  Lưu ý:

  Các hình vẽ hoàn toàn sử dụng thư viện của Processing.js

  Do vậy, sinh viên hãy vận dụng kiến thức trong bài học để vẽ hình cho project này.

Phần 3. Tài nguyên

Tài nguyên sẵn có bao gồm folder begin. Các bạn có thể download tại đây.
Ngoài ra các bạn có thể tham khảo code của các Challenge trong khoá học online trên KhanAcademy.

Challenge: Funky Frog
https://www.khanacademy.org/computing/computer-programming/programming/variables/pc/challenge-funky-frog

Challenge: Parting Clouds
https://www.khanacademy.org/computing/computer-programming/programming/animation-basics/pc/challenge-parting-clouds

Challenge: Exploding Sun
https://www.khanacademy.org/computing/computer-programming/programming/animation-basics/pc/challenge-exploding-sun

Object Inheritance

https://www.khanacademy.org/computing/computer-programming/programming/object-oriented/pt/object-inheritance

Phần 4. Hướng dẫn chi tiết

1. Tái cấu trúc lại code, chuyển code vẽ oto ra thành 1 hàm riêng tên là drawCar()
2. Thử gọi lại hàm drawCar() trong function draw xem oto có chạy được hay không
3. Thêm các tham số về màu sắc tốc độ cho drawCar(red, blue, green, x, y ….)
4. Gọi 3 hàm drawCar(red, blue , green, …) và truyền vào đó các tham số khác nhau để tạo ra 3 xe có màu sắc vị trị và thậm chí là tốc độ khác nhau
5. Viết lệnh kiểm tra điều kiện nếu 2 xe cùng 1 vị trí, thì tạo ra vụ nổ (cùng toạ độ, hoặc va chạm đầu đuôi, hoặc đi sát nhau…tức là chỉ cần có điểm va chạm thì sẽ tạo rao vụ nổ).

Giả xử mỗi xe sẽ có điểm xuất phát là (x,y) là vị trí để bắt đầu tính mốc vẽ các khung hình (thân xe, đầu xe, bánh xe). Mỗi xe giả xử có độ dài là 100 và độ rộng là 25. Vậy thì vùng không gian của xe là x + 100 và y + 25. Điều này có nghĩa là sẽ xảy ra va chạm nếu xe nào đó có toạ độ x rơi vào vùng an toàn của xe hiện tại.

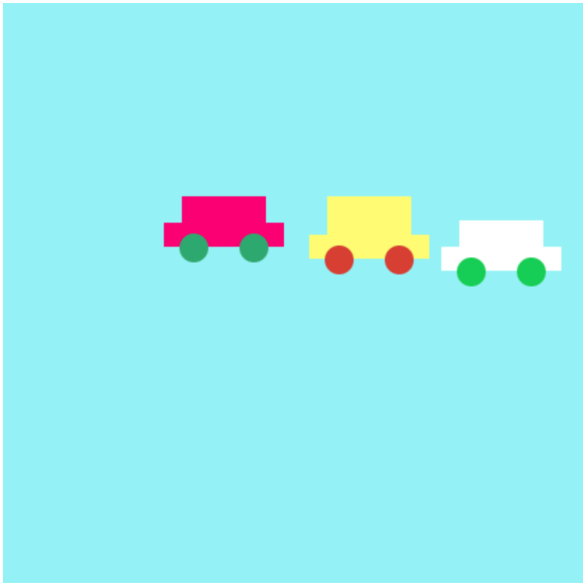
```
x = x + 1;
if(x==300) x = 10;

// draw the car body
car3_x = x + 120;
pointy = 10;
fill(255, 255, 115);
rect(car3_x, 200 + pointy, 100, 20);
rect(car3_x + 15, 178, 70, 40);

//va chạm
if(car3_x >= x && car3_x <= x + 100) {
  ellipse(x+62, 150, 70, 60);
}

// draw the wheels
fill(217, 65, 56);
ellipse(car3_x + 25, 221+ pointy, 24, 24);
ellipse(car3_x + 75, 221+ pointy, 24, 24);

// draw the car body
```



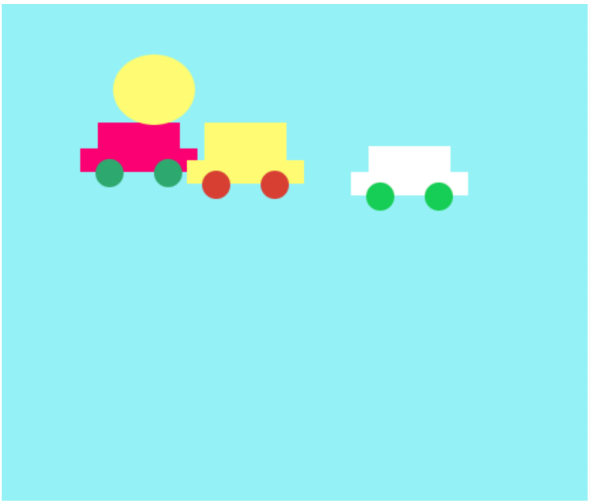
Ví dụ ta có 2 xe: xe 1 có toạ độ là: x=10, thì vùng an toàn của xe là x từ 10 tới 110 (100 + 110)
Vậy nếu xe thứ 2 có x nằm trong khoảng từ 10 tới 110 thì va chạm sẽ xảy ra.
Ví dụ đây là đoạn code mô phỏng đoạn xử lý va chạm mà có x thứ 2 là car3_x = x + 120; là nằm ngoài khoảng

Còn đoạn này thì car3_x = x + 90 là nằm trong khoảng an toàn của xe 1, nên va chạm xảy ra.

```
// draw the car body
car3_x = x + 90;
pointy = 10;
fill(255, 255, 115);
rect(car3_x, 200 + pointy, 100, 20);
rect(car3_x + 15, 178, 70, 40);

//va chạm
if(car3_x >= x && car3_x <= x + 100) {
  ellipse(x+62, 150, 70, 60);
}

// draw the wheels
fill(217, 65, 56);
ellipse(car3_x + 25, 221+ pointy, 24, 24);
ellipse(car3_x + 75, 221+ pointy, 24, 24);
```

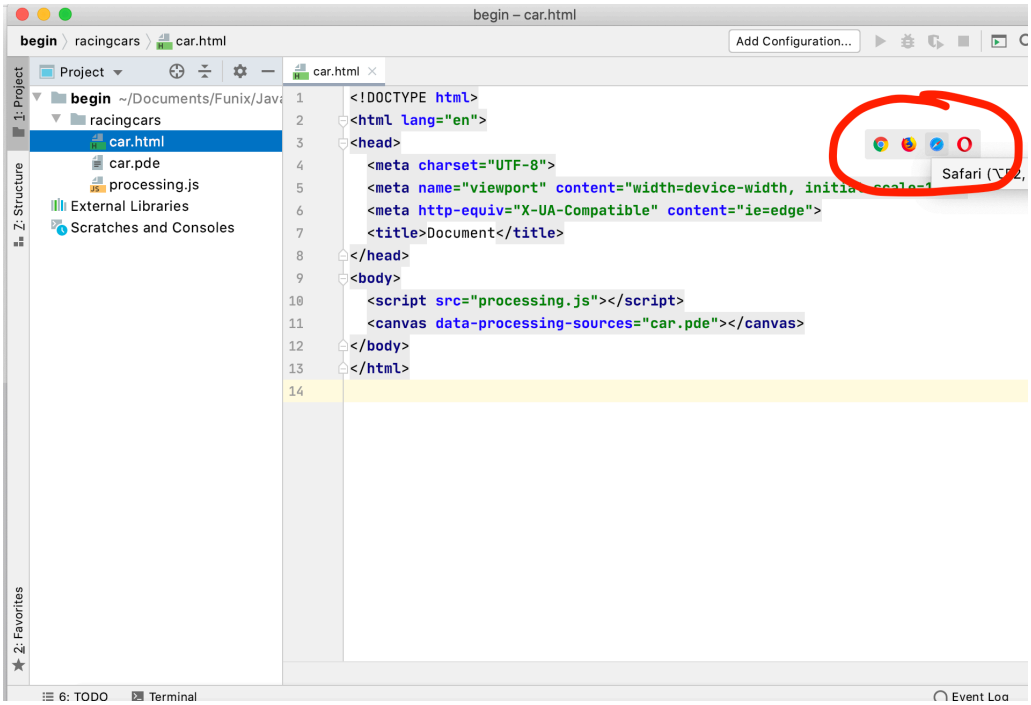


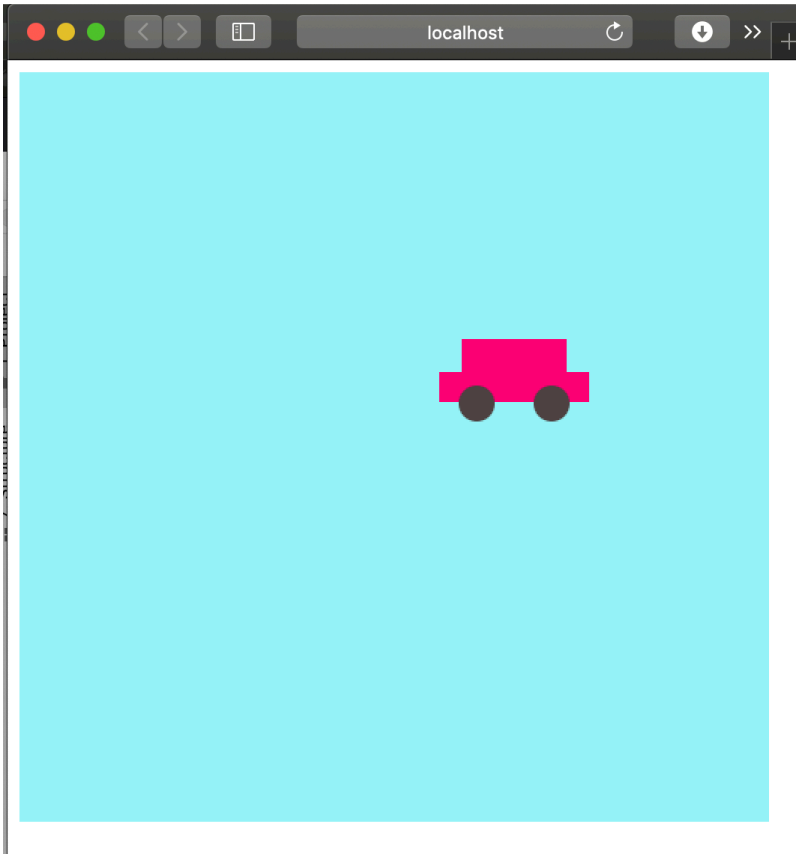
6. Tạo ra các chướng ngại vật xuất hiện tại vị trí bất kỳ
7. Viết lệnh kiểm tra vị trí của chướng ngại vật và xe cùng 1 chỗ thì tạo ra vụ nổ
8. Viết lệnh kiểm tra nếu có xe về tới vạch đích thì dừng cuộc chơi
9. Vẽ cảnh quan của trường đua bằng kiến thức của những bài về biến, hàm và animation.

Phần 5. Hướng dẫn chạy processing.js trên local

Để chạy processing.js trên máy của mình, các bạn làm theo các bước sau.

1. Vào trang processing.js download thư viện processing.js về máy. http://processingjs.org/download/
2. Vào mục Learning của trang processingjs.org để tham khảo các cách chạy khác nhau http://processingjs.org/learning/
3. Cài đặt WebStorm https://www.jetbrains.com/webstorm/
4. Kéo thả folder begin vào để chạy thử ứng dụng ban đầu
5. Mở file .html rồi chọn trình duyệt phù hợp để chạy.


begin – car.html
begin
racingcars
car.html
Add Configuration...
Project
car.html
begin ~/Documents/Funix/Java
racingcars
car.html
car.pde
processing.js
External Libraries
Scratches and Consoles
1: Project
7: Structure
2: Favorites
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initia
6 <meta http-equiv="X-UA-Compatible" content="ie=edge">
7 <title>Document</title>
8 </head>
9 <body>
10 <script src="processing.js"></script>
11 <canvas data-processing-sources="car.pde"></canvas>
12 </body>
13 </html>
14
Safari (⌥⌘2,
6: TODO
Terminal
Event Log


localhost