CP317 Fall 2016

Assignment 1

Due: Tuesday October 4 at 11:30PM

The purpose of this assignment is to modify the *simple canvas game* published here. Based on that game, make a game subject to the following requirements

- The hero's location is at the middle point of the left edge of the screen. This position cannot be changed during the game.
- Every three seconds a goblin appears at a random point on the right edge of the screen.
- The goblins move horizontally toward the left edge of the screen.
- A *mousedown* event on a point P on the screen triggers a bullet firing from the center of the hero to the direction of P, provided P is not in the hero's icon. You should programmatically destroy the bullet when it hits the screen edge, for otherwise keeping track of a large number of bullets will slow down your game and possibly crash it. A goblin is shot (destroyed) when it is hit by a bullet. The number of goblins caught should be updated. When a goblin is shot, a distinct sound should be played.
- When a goblin reaches the left edge of the screen, it is considered *escaped*. The number of goblins escaped should be updated. When a goblin escapes, a distinct sound (different from the goblin shot sound) should be played.
- If goblin collides with the hero, the goblin is considered shot and should be destroyed.
- The canvas' width and height should be the entirety of the browser's window.
- The background image should cover the whole canvas.
- Use localStorage to store the numbers of goblins caught and escaped. These numbers should persist and be displayed across launches of the game.
- Add background sound.

Bonus points:

- Can you think of, and implement, a new feature to make the game more interesting? A new feature should add to the game, it should not replace a required feature. If you add bonus features, mention them in the submission for otherwise the instructor may not be aware of them.

Programming hints

You may use any editor to write Javascript programs. I use Notepad++ (free).

Never hard wire a literal in code. Define a constant for it, and use the constant in the code.

The size of the canvas should never be hard-coded. It should depend only on the screen size of the browser window

```
canvas.width = window.innerWidth;
canvas.height = window.innerHeight;
```
Similarly, the size of the sprites should be declared as constants in the Javascript files. If you need to change the size, you will only need to change this constant declaration and not, say, 100 lines of codes where the size is used.

Another great HTML5 game sample is given here. For the game, press Space to shoot, left-arrow and right-arrow keys are for moving the player. See how collisions are detected, and sounds are played.

For most web projects, the Javascript files are put in a folder typically named "js", the images in "images", sounds in "sounds", etc.

Javascript functions/features you may need to know (look them up in http://www.w3schools.com/)

- onload
- localStorage
- addEventListener
    - "mousedown", "click", etc
- Image
- Audio
- drawImage
- setInterval

Developer tools for Chrome

Chrome provides tools for developing Javascript apps (Settings → Tools → Developer Tools, or Control-Shift-I). There is a console tab that provides messages from the browser. You can log (print out) messages to the console with the Javascript with `console.log`. The messages do not go to the rendered web page. This feature is useful for debugging. Example:

```
canvas.width = window.innerWidth;
canvas.height = window.innerHeight;
console.log('canvas.width=' +  canvas.width + ' canvas.height=' +
canvas.height);
```
Your submission will be tested on Chrome.

The game loop

We now discuss the design of a computer game. Every game has two main functions: `update` and `render`.

The function `update`

- updates the position of very sprite in the game,

- determines whether two sprites collide (say, if a bullet collides with a monster, then the monster should die and the bullet should be removed from the game)
- removes obsolete/dead/unused sprites from the game.

The function `render`

- draws the sprites on the canvas.

Typically the function `update` is called every 1/30-th of a second (or at the refresh rate of the computer monitor), then the function `render` is called. The function `requestAnimationFrame` will execute a given function at the refresh rate. The function `setInterval` will execute a given function in an interval in milliseconds.

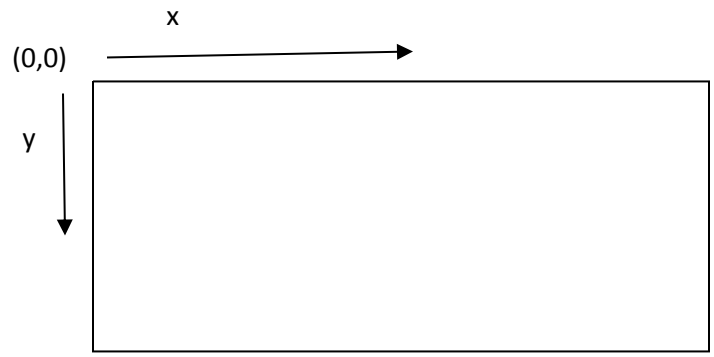
Detecting sprite collisions

A sprite is defined be a center (which is a point with a x-coordinate and an y-coordinate) and a box surrounding it. The box is defined by its width and height. Two sprites collide if their boxes overlap.

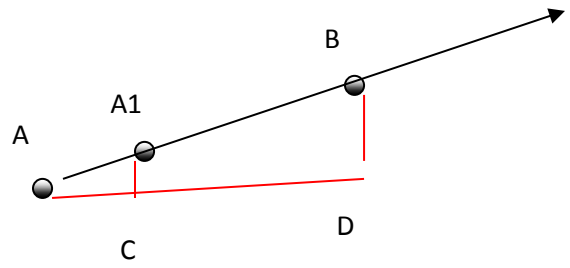
```
function collides(a, b) {
  return a.x < b.x + b.width &&
         a.x + a.width > b.x &&
         a.y < b.y + b.height &&
         a.y + a.height > b.y;
}
```

Computing the trajectory of the bullets

The canvas origin is the top left corner.



Suppose the hero's location is at point A, and the *mousedown* event's location is a point B. A bullet will have to be created at point A, then it travels along the line A-B, until it exits the canvas. The bullet may not stop at point B.



For each call to the function *update*, we have to calculate the (new) position, say at point A1, of the bullet along the line A-B. We will have to use the trigonometry knowledge we learned in high school. We need to calculate the sides of the triangle A-C-A1 . We know the lengths of the three sides of the triangle A-D-B because we know points A and B (and therefore D).

AD = B.x – A.x (where x denotes the x-value of the point)
BD= B.y – A.y (where y denotes the y-value of the point)

The direction (angle) of the line AB is:

dir = Math.atan(BD/AD)

where atan is the arctan function in Javascript. We may define *speed* to be the speed of the bullet. This is a constant and you may experiment with it to arrive at an appropriate value of *speed* for your game. You may start with, say, *speed* = 10. Let *xSpeed* be the speed of the bullet along the x-axis, and define *ySpeed* similarly. We have

*xSpeed = sin(dir) * speed*
*ySpeed = cosin(dir) * speed*

Then the new position of the bullet, at the point A1 is :

*bullet.x = bullet.x + bullet.xSpeed * bullet.speed;*
*bullet.y = bullet.y + bullet.ySpeed * bullet.speed;*

All of this is explained in detail in this article.

Other requirements

You may not use any Javascript framework to do this assignment. You must base your code on the *simple canvas game*.
You must compute the positions of the sprites using the above equations.

Submission instructions

Put the names of your group members in the header of the comments. Put all your files in a folder. Name the folder with your group number and the assignment number. Compress (zip) this folder and submit it at mylearningspace.

Failure to follow the above instruction will result in mark reductions.

A screen shot of the game is shown below