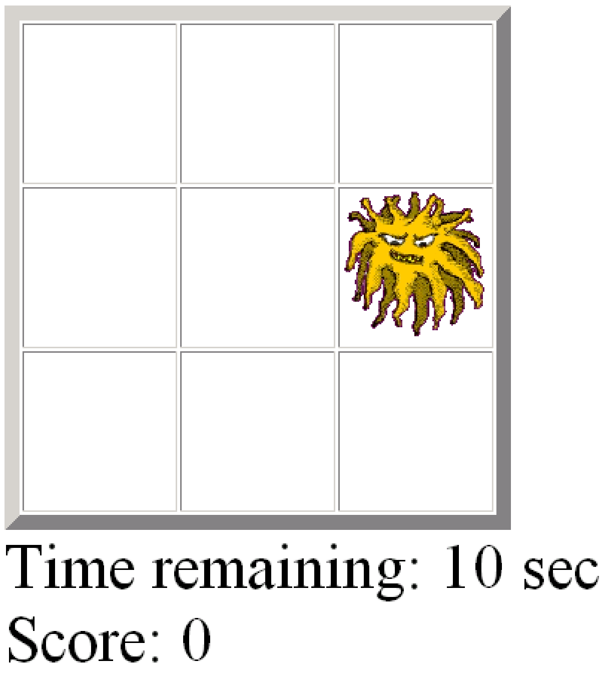
# COMP4021 Lab 1
Hammer Game

## Overview

- In this lab, you need to create a simple "hammer" game by writing your own JavaScript code



Time remaining: 10 sec
Score: 0

- The game runs as follows:
  1. When the game is loaded, a clock counts down from 10 seconds to zero
  2. A monster appears randomly
  3. It moves to a new position after a certain period of time
  4. Players have to press the corresponding key to hit the monster before it moves
  5. If the player hits the monster, the score increases and the monster moves immediately
  6. When time is up, the game is over and everything freezes
- When playing the game, the corresponding keys for the nine boxes are shown below:

| | | |
|---|---|---|
| Q | W | E |
| A | S | D |
| Z | X | C |

- We will demonstrate the game during the lab session

## Getting Started

- An HTML file `hammer.html` is given to you as the starting code of the program here
  (right click on the link, select 'Save Target As', and save it to your computer)
- Two sample pictures are given to you here: the empty background, empty.png, and monster, monster.png
- You can use any text editor such as Notepad to build your program
  - Alternatively you can use TextPad, which is available in the CS labs, to edit your program

## Programming Overview

### Complete five predefined JavaScript functions

- `game_start()`
  - This is called by the `onload` attribute in the `<body>` tag
  - It starts the game by calling the count down function and monster relocating function
- `count_down()`
  - If the time limit of the game is 10 seconds, then this function will be called 10 times to update the clock
  - When the count down is finished, it calls the game over function
- `relocate()`
  - This moves the monster to a new position. Do it again after a certain period of time
- `keyboard_event()`
  - Collect player input
  - If the player hits the monster, update his/her score and relocate the monster
- `game_over()`
  - Stop everything and show a message

### Create two JavaScript timers

- `count_down_timer`
  - Count down from 10 to 0 in the game
- `moving_timer`
  - Handle the monster relocation after a certain period of time

### Create four variables at the beginning

- `var score = 0;`
  - The score of the player
- `var time_remaining = 10;`
  - Store the time remaining
- `var monster_position = 0;`
  - Store the current position of the monster (a number from 0 to 8)
- `var finished = false;`
  - Indicate whether the game is finished or not

### Example JavaScript programs

- Count down - it shows a timer counting down from 10 sec to 0 sec
- Key press - it asks you to press a key, and then it shows what key you have pressed
- Image index - it shows a monster appearing in a specific position when you press a number from 0 to 8

  Remember you can always look at the source code by selecting View Source (Internet Explorer) or Ctrl-U (Firefox)

## Programming Procedure

## 1. Start the game

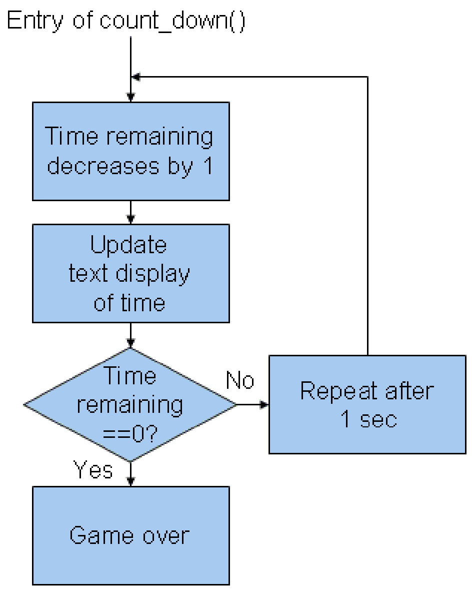
- In the starting code, we have set `game_start()` to be the first function to run when the page is loaded

```
<body onload="game_start()">
```

- `game_start()` should call `count_down()` after 1 second
- Also, it will call `relocate()` without delay, so that the monster starts to move immediately

## 2. Create a 10 seconds count down timer

- `count_down()` works as follows:



- In `count_down()`, we need to do three things:
  1. Decrease the time remaining by 1

```
time_remaining = time_remaining - 1;
```

  2. Update the timer display

```
var timer_element = document.getElementById("timer_text");
timer_element.innerHTML = "Time remaining: " + time_remaining + " sec";
```

  3. If `time_remaining` equals 0, the game is over; otherwise, call this function again after a second

```
if (time_remaining == 0)
    game_over();
else
    count_down_timer = setTimeout("count_down()", 1000);
```

## 3. Finish the game

- In `game_over()`, we need to stop the timer and print an appropriate message

```
clearTimeout(moving_timer);
alert("Times up!!!");
```

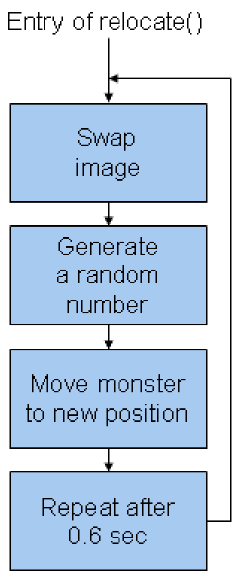
- Also, set the `finished` variable to true

## 4. Move the monster to a new position

- `relocate()` will be called in these situations

1. When the game just begins, it is called by `game_start()`
2. After the monster has appeared in one location for a certain period of time, it relocates
3. After the player hits the monster, the monster relocates

- `relocate()` works as follows:



- Before the monster moves to a new location, we have to clear the image first
- We do this using a trick, we change the image by changing the `.src` value
- You have already learnt how to generate a random number
- Using the same technique we generate a random from 0 to 8 and store the new position in the variable `monster_position`
- For the name of each box, you can refer to the table below:

| | | |
|---|---|---|
| document.images[0] | document.images[1] | document.images[2] |
| document.images[3] | document.images[4] | document.images[5] |
| document.images[6] | document.images[7] | document.images[8] |

- Then at the new position we set the picture to "monster.png"

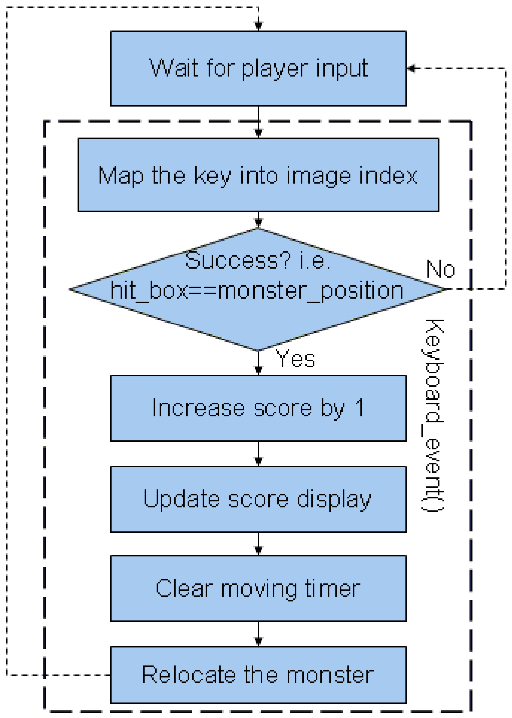
```
document.images[monster_position].src = 'monster.png';
```

- So now the monster has moved
- However, in order to make the game more interesting, the monster should automatically jump to another location after a certain period of time, say, 0.6 sec
- So, at the end of the function, we use `setTimeout` to call `relocate()` again after a delay

```
moving_timer = setTimeout("relocate()", 600);
```

## 5. Hit the monster box

- We have to work on `keyboard_event()` so that the game responds to the player
- It works as follows:



- The body should respond to the keyboard event only when the game is not finished

```
<body onKeyDown="if (!finished) keyboard_event(event)">
```

- `keyboard_event()` should carry out two major tasks:
  1. Transform the player's input into the corresponding image index
     - Extract the pressed key from `event.keyCode`

     ```
     var pressed_key = String.fromCharCode(event.keyCode);
     ```

     - Transform the pressed key into image index and then store the result in a new variable `hit_box`

     ```
     var hit_box;
     switch(pressed_key)
     {
         case 'Q': hit_box=0; break;
         case 'W': hit_box=1; break;
         . . . // And so on
     }
     ```

  2. If the player presses the correct key, update the score and move the monster to a new position

     ```
     if (monster_position == hit_box) {
         // Update score and relocate monster
         . . .
     }
     ```

     - Increase the value of the variable `score` by 1
     - Update the score display, we can use `.innerHTML` to do this
     - Move the monster immediately by calling `relocate()`; but, before doing this, we have to clear the `moving_timer` first so there is no confusion

     ```
     clearTimeout(moving_timer);
     ```

## 6. Add sound

- To make the game funnier, we can add a sound when we hit the monster using the `<audio>` tag
- You can get sample sound files ouch.mp3 for hit and boo.mp3 for miss

  Mostly we will use wav file, here mp3 file is for IE browser. Chrome and Firefox can support both files.
- To achieve that, we have to add the following syntax inside the `<body>...</body>` area of the HTML:

```
        <audio src="./hammer_files/boo.mp3" type="audio/mpeg" width="0" height="0" id="boo" > </audio>
        <audio src="./hammer_files/ouch.mp3" type="audio/mpeg" width="0" height="0" id="ouch"> </audio>
```

- Remember to add the JavaScript code `pause()` and `play()` at the right place
- So when the monster is hit, the sound will be played immediately
- If the monster is hit again, the playing sound will be stopped and played again

## Submission

- You do not need to submit the lab work