# NM2207

Session 09 Challenges

Before attempting the challenges, you are expected to have watched and coded along with the Lecture videos. A tutorial is meant to practice the skills presented in the video lecture, and show you more applications of it. Tutors will explain the challenges and answer the questions you may have**.**

**The challenges are due to be completed at the end of class each week for full credit which is also attendance. Submitting by midnight of the same day (ie Thursday night) accounts for half the credit.**

# Overview of what we will do today:

* Explore the model view controller logic
* Practice using the setInterval
* Practice creating new objects at each interval

**Part 1 (40 minutes)**

**Summary of learnings**

* Revising model view controller logic
* Completing an update function

**Warm up (40 minutes)**

The Overall Goal of this task set is to explore the code and answer some questions.

* 1. Watch the video
  2. Answer the following questions as (a) a description based on what you saw in the video, and (b) function names based on your exploration of code:

Table 1

|  |  |  |
| --- | --- | --- |
| Question | What happens | Line that makes it happen |
| What should happen when the page is loaded | A message should be written on the console and the game should start | function startGame() { //when the page is loaded, what should happen?  myGameArea.start();  user.name = window.prompt("Enter your name"); |
| Example the component function. Where is it called? What does it return? | Component is called when drawPiece() is called. But drawPiece isn’t called YET. Until we link it to the NEXT button.  Component will return a rectangle with the inputted attributes | function component(width, height, color, x, y) {  var rectangl = {}  rectangl.width = width;  rectangl.height = height;    rectangl.x = x;  rectangl.y = y;  rectangl.shape = paper.rect()  rectangl.shape.attr({x:rectangl.x, y:rectangl.y, width:rectangl.width, height:rectangl.height,fill: color});  rectangl.update = function() {  rectangl.shape.attr({x:rectangl.x, y:rectangl.y, width:rectangl.width, height:rectangl.height,fill: color});  }  return rectangl;    } |
| When do you first see a rectangle on the screen? | When you load the screen (The white background) | Onload="startGame()" |
| What should happen when the game starts? | A prompt should ask for the user name. The value of frameNo (which is 0) is printed to the console | startGame()  myGameArea.start() |
| What should happen when the “next” button is clicked for the first time? | A green rectangle should be created in the far corner of the paper | drawPiece() |
| What should happen when the “next” button is clicked each subsequent time? | The rectangle moves by a frameNo | updatePiece() |
| What should happen when the user clicks on the rectangle? | It will input their score | User.updateScore |
| When does the game end? | After frameNo >= 10 | myGameArea.stop()  checkEndGame() |
| What happens when the game ends? | They will prompt you with how many frameNo you get | checkEndGame()  updateGameArea() |

* 1. Now, based on your answers to Table 1, update the checkEndGame() function and the updateGameArea() function. Where should the checkEndGame() function be called?

**Part 2 (1h)**

**Summary of learnings**

* **Practice using the setInterval**

The Overall Goal of this task set is to stop using the “next” button and use setInterval instead. Specifically:

2.1 Create a property of myGameArea called timer which will be used to call the setInterval method and the function to repeat at each interval of 100 milliseconds. Call it each time the page is loaded.

2.1 Use the clearInterval function on the timer each time the game ends. Use the myGameArea.stop() function to do this, and update the checkEndGame() function accordingly.

**Bonus**

**Part 3: Spawning new objects**

* 1. See if you can end the game at every 30 frames instead of 10.
  2. See if you put the “green” object in a list instead. Use something like this:

**var myObstacles = [];**

**myObstacles.push(component(10, height, "green", x, 0));**

* 1. For every fifth frame, see if you can “spawn” a new component by using the above code.
  2. But then, remember that you need to move all the obstacles in each updatePiece function call.