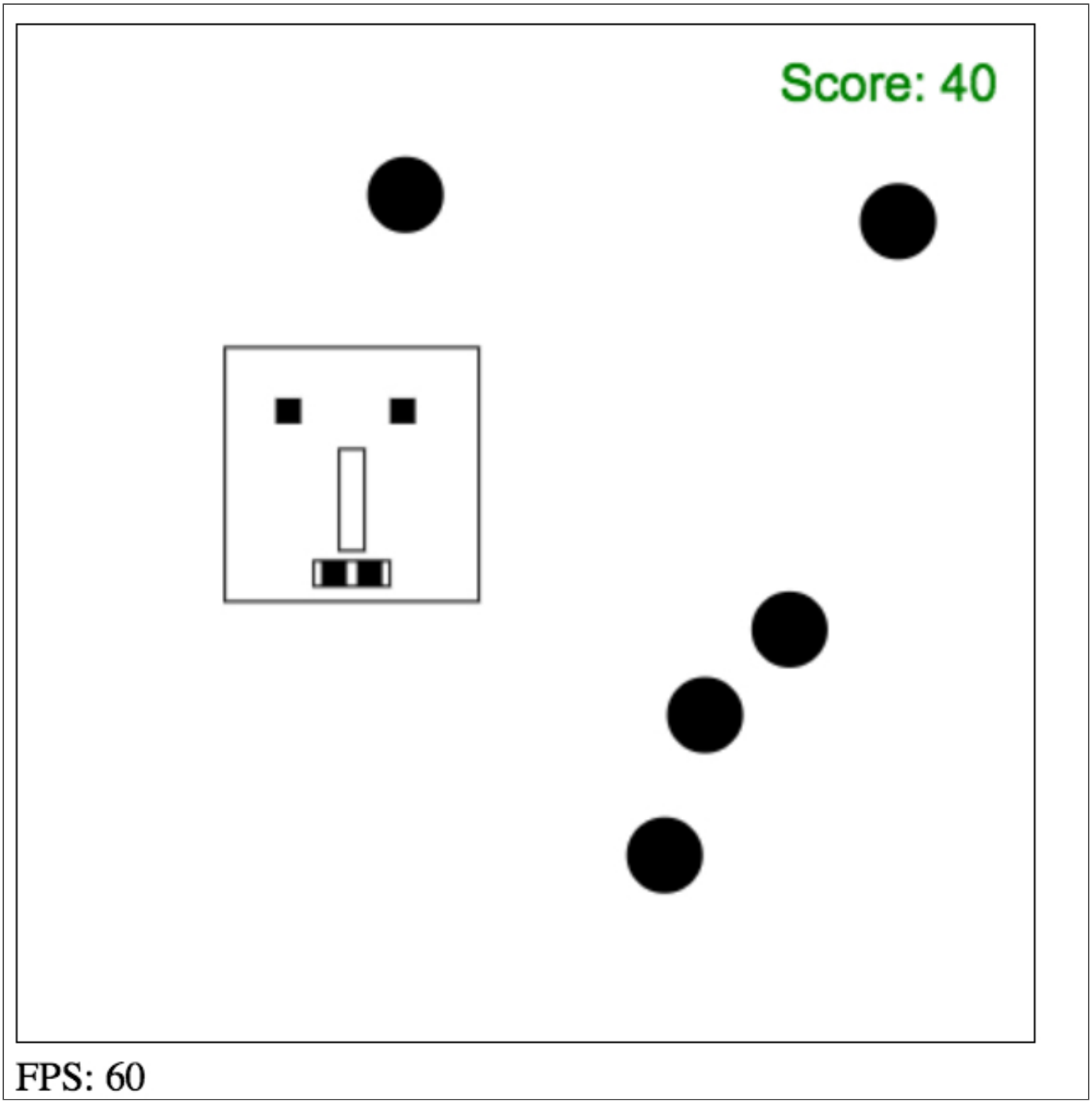
Game and object states: start menu, game over menu, etc.

OBJECT STATES

Now, we can make things more exciting by making something happen when a collision occurs - maybe kill the player, or kill the balls? Or increment your score. Usually, we add extra properties to the player and enemies. For example, a boolean `dead` property that will say if an object is dead or alive: if a ball is marked "dead", do not draw it! If all balls are dead: go to the next level with more balls, faster balls, etc...

Let's try adding a `dead` property to balls and stop drawing them if they are dead! We also test when all balls are dead, in which case we recreate them and add one more ball. We update the score each time the monster eats a ball. And finally we added a test in the`createBalls` function to ensure that no balls are created on the monster.

Try this jsBin!



Source code extract:

```
function updateBalls(delta) {
   // for each ball in the array
   var allBallDead = true;
   for(var i=0; i < ballArray.length; i++) {
     var ball = ballArray[i];
     if(ball.dead) continue; // do nothing if the ball is
```

```
    dead
10.     // if we are here: the ball is not dead
        allBallDead = false;
        // 1) move the ball
        ball.move();
        // 2) test if the ball collides with a wall
        testCollisionWithWalls(ball);

        // Test if the monster collides
20.     if(circRectsOverlap(monster.x,monster.y,
                            monster.width,monster.height,
                            ball.x, ball.y,ball.radius)) {
           //change the color of the ball
           ball.color = 'red';
           ball.dead = true;
           // Here, a sound effect would greatly improve
           // the experience!
30.        currentScore+= 1;
        }
        // 3) draw the ball
        ball.draw();
     }
     if(allBallDead) {
        // reset all balls, create more balls each time
        // as a way of increasing the difficulty
        // in a real game: change the level, play nice music!
        nbBalls++;
        createBalls(nbBalls);
     }
}
```

GAME STATES: MENUS, HIGH SCORE TABLES ETC...

In this example we will see how we can use a global `gameState`variable for managing the lifecycle of your game. Usually we've got a main menu, with a "start new game" option, then we play the game, and when we die we can see a game over screen, etc...

Ah!... you think that the game is too easy? Let's reverse the game: now you must survive

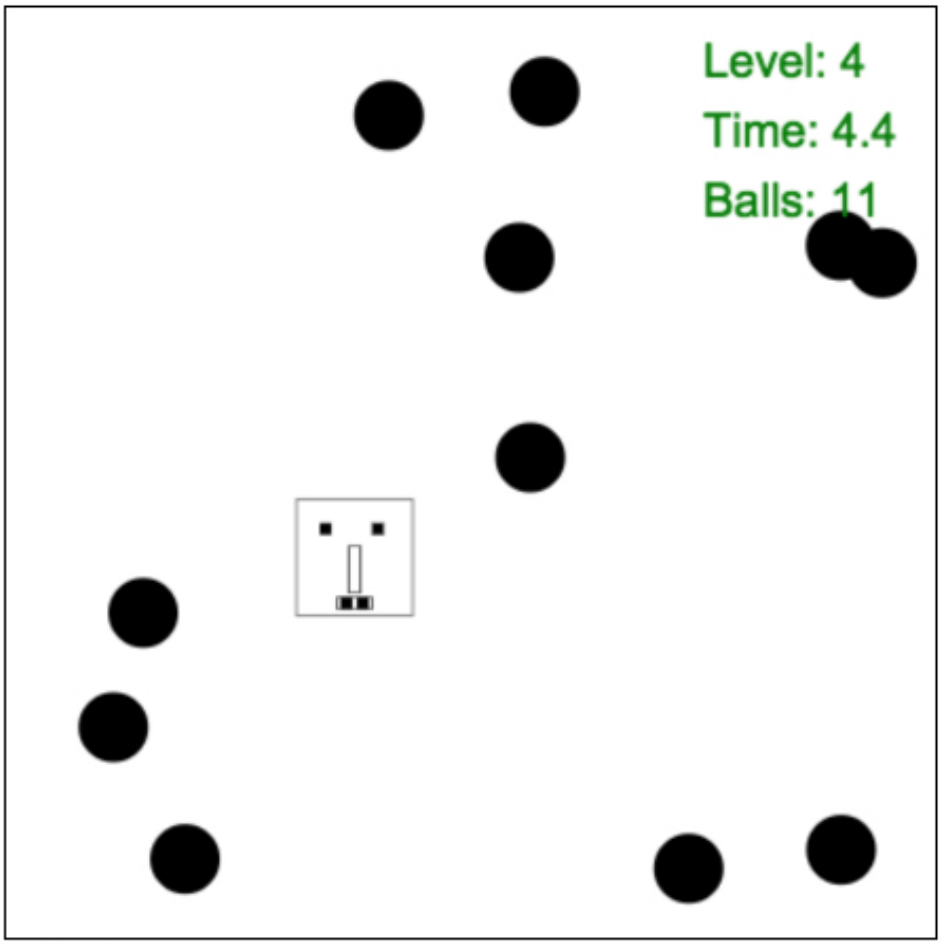
without being touched by a ball!!!

Every 5 seconds a next level starts, and the set of balls is re-created, this time with two additional balls. How long can you survive?

Try this JsBin, then look at the source code. Start from the `mainloop`!

```
currentGameState = gameStates.gameOver:
```


```
currentGameState = gameStates.gamerunning:
```



Game state management in the JavaScript code:

```
...
// game states
var gameStates = {
  mainMenu: 0,
  gameRunning: 1,
  gameOver: 2
};

var currentGameState =gameStates.gameRunning;
var currentLevel = 1;
var TIME_BETWEEN_LEVELS = 5000; // 5 seconds
var currentLevelTime = TIME_BETWEEN_LEVELS;
...
var mainLoop = function (time) {
  ...
```

```
    // number of ms since last frame draw
    delta = timer(time);

    // Clear the canvas
    clearCanvas();

    // monster.dead is set to true in updateBalls when there
    // is a collision
    if (monster.dead) {
        currentGameState = gameStates.gameOver;
    }

    switch (currentGameState) {
        case gameStates.gameRunning:
          // draw the monster
          drawMyMonster(monster.x, monster.y);

          // Check inputs and move the monster
          updateMonsterPosition(delta);

          // update and draw balls
          updateBalls(delta);

          // display Score
          displayScore();

          // decrease currentLevelTime. Survive 5s per level
          // When < 0 go to next level
        currentLevelTime -= delta;

        if (currentLevelTime < 0) {
            goToNextLevel();
        }
        break;
      case gameStates.mainMenu:
        // TO DO! We could have a main menu with high scores
    etc.
        break;
      case gameStates.gameOver:
        ctx.fillText("GAME OVER", 50, 100);
```

```
            ctx.fillText("Press SPACE to start again", 50, 150);
            ctx.fillText("Move with arrow keys",50, 200);
            ctx.fillText("Survive 5 seconds for next
level", 50, 250);
58.
            if (inputStates.space) {
                startNewGame();
            }
            break;
        }
         ...
    };
...
```

And below are the functions for starting a new level, starting a new game, and the `updateBalls` function that can make the player dead and change the current game state to GameOver:

```
function startNewGame() {
    monster.dead = false;
    currentLevelTime = 5000;
    currentLevel = 1;
    nbBalls = 5;
    createBalls(nbBalls);
    currentGameState =gameStates.gameRunning;
}

10. function goToNextLevel() {
    // reset time available for next level
    // 5 seconds in this example
   currentLevelTime = 5000;
   currentLevel++;
   // Add two balls per level
   nbBalls += 2;
   createBalls(nbBalls);
}

function updateBalls(delta) {
```

```
        // Move and draw each ball, test collisions,
22.     for (var i = 0; i < ballArray.length;i++) {
            ...
            // Test if the monster collides
25.         if (circRectsOverlap(monster.x,monster.y,
                                 monster.width,monster.height,
                                 ball.x, ball.y,ball.radius)) {

                //change the color of the ball
                ball.color = 'red';
                monster.dead = true;
                // Here, a sound effect greatly improves
                // the experience!
                plopSound.play();
35.         }

            // 3) draw the ball
            ball.draw();
        }
    }
```