Universidade de Brasília – Faculdade UnB Gama
Disciplina: Técnicas de Programação
Professor: Maurício Serrano
Período: 2º/2015
Data de entrega: 24 de Agosto de 2015
Atualizado dia: 19 de Outubro de 2015
Estudantes:

11/0014863 - *Kleber B. Moreira*
11/0017561 - *Mateus M. F. Mendonça*
11/0017692 - *Matheus M. Nascimento*
11/0064879 - *Nicolas Boarin*

Stylesheet - Grandes Pontos

1. Comments

Comments must obey the following pattern:

1.a) One-line comments must be written using a double slash '//' before the comment, followed by a space ' ' and a capital letter. Do not finish it with a period;

```
// For each disc in discs, draw itself
```

1.b) Comments in blocks must be written between the '/*' and '*/'. For each new line, there must have an asterisk aligned with the previous one;

```
/**
 * For each pawn, fetch its sprite and push the object into the array.
 * Its y coordinate is changed to be drawn on top of the previous one.
 */
```

1.c) Comments which precede methods must be written in blocks explaining its use and its return:

```
/**
 * Window On Click adjust the mouseClick to gridClick and calls resolveTurn()
 * @param mouseClick
 */
window.onclick = function(mouseClick){

  // Object mouse containing the coordinate (x,y) of the event
  var mousePosition = {
    x: mouseClick.pageX,
    y: mouseClick.pageY
  };
  //----
}
```

1.d) Comments explaining one line of the code must precede the line;

```
// Create and Seal object to prevent properties addition
var discObject = new GameObject(discSprite, positionCoordinates, scale);
Object.seal(discObject);
```

1.e) Comments exceeding 80 characters must be broken into multiple lines following the block pattern;

```
/**
 * For each pawn, fetch its sprite and push the object into the array.
 * Its y coordinate is changed to be drawn on top of the previous one.
 */
```

2. Types and Names

2.a) Names of Classes, Attributes, and Methods must follow the CamelCase pattern, which consists in capitalizing the initial letters of each word.

2.b) Classes names initiates with capital case.

```
var pawnObject = new GameObject(sprite, positionCoordinates, scale);
```

2.c) Attributes names initiates with lower case.

```
var isValid = false;
```

2.d) Methods names initiates with lower case.

```
GameObject.prototype.move = function(positionCoordinates) {
    this._sprite.positionCoordinates = positionCoordinates;
}
```

2.e) Constants names must be capitalized and use underscore between names.

```
// Number of discs' colors
const NUMBER_DISC_COLORS = 7;
```

2.f) Only one variable must be declared per line.

```
var canvas          = null,
    context         = null,
    offsetWidth     = 20,
    offsetHeight    = 20;
```

3. Strings

3.a) Strings must be written between double quotes ‘ ” ‘;

```
const GREEN_DISC_NAME = "Multi/disc_green.png";
```

4. Indentation

4.a) The indentation must be written by two (2) spaces or an equivalent tab.

```
function validateDisc(discCount, discColor) {

--var isValid = false;

--// Discs white and black are considered special discs
--if (discColor === WHITE || discColor === BLACK) {
----isValid = (discCount[discColor] < DISC_SPECIAL_LIMIT)
--} else {
----isValid = (discCount[discColor] < DISC_LIMIT)
--}

--return isValid;
}
```

5. Braces

5.a) Opening braces must be used on the same line of the block structure, after an empty space;

5.b) Closing braces must be written aligned with the statement which opened the block;

```
function validateDisc(discCount, discColor) {
|
```

```
| var isValid = false;
|
| // Discs white and black are considered special discs
| if (discColor === WHITE || discColor === BLACK) {
| | isValid = (discCount[discColor] < DISC_SPECIAL_LIMIT)
| } else {
| | isValid = (discCount[discColor] < DISC_LIMIT)
| }
|
| return isValid;
}
```

6. Classes

Classes must be according to the following model:

6.a) There must have a blak line after the class signature.

```
/**
 * The constructor to create a GameObject
 *
 * @param
 * Sprite as defined in Atlas.js
 * positionCoordinates (x,y)
 * scale (width, height)
 */
var GameObject = function (sprite, positionCoordinates, scale) {
    this._sprite = {};
    try {
        this._sprite.name = sprite.name;
        this._sprite.sourceCoordinates = sprite.sourceCoordinates;
        this._sprite.dimensions = sprite.dimensions;
        this._sprite.positionCoordinates = positionCoordinates;
        this._sprite.scale = scale;
    }
    catch (errorSprite) {
        alert(errorSprite);
    }
};

```

6.b) Prototypes must be used as inheritance:

```
function Person( name ) {
  var name = name;
}

PhysicalPerson.prototype = new Person();

var physicalPerson = new PhysicalPerson( "Nome da Pessoa" );
console.log( physicalPerson.name ); // Nome da Pessoa
```

7. Control Structure: `if`

7.a) There must have a blank space immediately after the keyword if.

7.b) There must have a blank space between the parentheses and the `if` condition.

7.c) The comparison operators must have a blank space immediately before and after.

7.d) If the line exceeds 80 characters, it must be broken into multiple lines and the condition must be aligned with the

previous one.

7.e) The body must be written between braces, even when it is a single line.

7.f) Blocks of `Else if` or `else` must be initiated on the same line of the closing braces.

7.g) There must have an `else` block after all `if` or `else if` block.

```
/**
 * Is Inside Board checks whether the click was in the board
 * @param gridClick
 * @return insideBoard
 */
function isInsideBoard(gridClick) {
  var insideBoard = false;

  if (gridClick.x < gridConfiguration.rows &&
      gridClick.y < gridConfiguration.cols &&
      gridClick.x >= 0 &&
      gridClick.y >= 0) {
    insideBoard = true;
  }
  else {
    insideBoard = false;
  }

  return insideBoard;
}
```

8. Control structure: `while`

8.a) `While` follows the `if` pattern:

```
while ( count < numElements ) {
  sumElements = sumElements + 1;
}
```

9. Control structure: `for`

9.a) The `for` structure follows the `if` pattern.

9.b) Semicolon must be placed immediately after the assignment and the comparison, and an empty space after it.

```
// Fill the board with random discs along the x and y axis (rows and columns)
for (var discX = 0; discX < NUMBER_DISC_ROW; discX++) {
  for (var discY = 0; discY < NUMBER_DISC_COL; discY++) {
    // ---
    var positionCoordinates = {
        x:((discX * DISC_DIMENSION.HEIGHT) + BOADR_OFFSET.X),
        y:((discY * DISC_DIMENSION.WIDTH) + BOADR_OFFSET.Y)
    }
    // ---
  }
}
```

10. Control structure: `switch`

a. Defaults and cases must not be indented;

10.b) Case blocks and default blocks must be indented.

10.c) `break` must be indented as well.

```
function fetchDisc(discColor) {
  var discSpriteName = "";
  switch (discColor) {
    case GREEN:
      discSpriteName = GREEN_DISC_NAME;
      break;

    case BLUE:
      discSpriteName = BLUE_DISC_NAME;
      break;

    case RED:
      discSpriteName = RED_DISC_NAME;
      break;

    case PURPLE:
      discSpriteName = PURPLE_DISC_NAME;
      break;

    case YELLOW:
      discSpriteName = YELLOW_DISC_NAME;
      break;

    case WHITE:
      discSpriteName = WHITE_DISC_NAME;
      break;

    case BLACK:
      discSpriteName = BLACK_DISC_NAME;
      break;

    default:
      // Should never be reached. There are only seven Discs's colors.
  }
  var discSprite = ATLAS.fetchSprite(discSpriteName);
  return discSprite;
}
```

11. Treatment of Exceptions

11.a) The `try`, `catch`, and `finally` blocks should be on their own line, after each closing braces.

```
try {
  this._sprite.name = sprite.name;
  this._sprite.sourceCoordinates = sprite.sourceCoordinates;
  this._sprite.dimensions = sprite.dimensions;
  this._sprite.positionCoordinates = positionCoordinates;
  this._sprite.scale = scale;
}
catch (errorSprite) {
  alert(errorSprite);
}
```

12. Coding language

The coding language for programming techniques applying is English.