PROJECT DESCRIPTION

OBJECT-ORIENTED PROGRAMMING FOR GAME DEVELOPMENT WITH P5.JS

PREPARED FOR: PROCESSING FOUNDATION
PREPARED BY: JUAN OLAYA

INTRODUCTION *what you're going to do*

This document describes the project **Object-Oriented Programming for Game Development with p5.js,** this is an online book that explains how to develop 2D video games using the JavaScript library p5.js. It has 20 lessons that shows examples focus on game development such as: player movements, collision detection, gravity simulation, jumps on platforms and shoot bullets.

The beginning of the book shows how to use a basic class with its parts, How to create an instance of a class and how to invoke its methods. Then, it is presented how to use polymorphism and inheritance, which are especially useful to display different video game characters with attributes and methods in common. At the end of the book, the collision detection algorithms are used to develop different video games mechanics.

In this online book, each of 20 lessons has:

- A text that describes the subject
- An example in p5.js with its documentation code. In order to identify which part of the script code is related to the class code, the code documentation has been divided by different colors in the background, making easily visible the structure of the code and easily understandable the links between the script and classes (as can be seen in the Lesson 1 and Lesson 2).
- A class diagram that describe the attributes and methods of each class and its relationship with the main script.
- And also, an embedded sketch running on OpenProcessing.

TO DO *how you're going to do it and why you're qualified for the work*

For this project I am going to use the curriculum, products and knowledge that I have built and acquired working with Processing and p5.js as university educator since 2016. I am Computer Engineering and I am currently teaching subjects such as: Introduction to Programming, Object-Oriented Programming, Data Structures and Software Engineering in the bachelor Digital Arts Engineering in the university Escuela de Artes y Letras (School of Arts and Letters) in Bogotá, Colombia.

In fact, I have tested the curriculum that this project proposes with my students. The following video games have been published on itch.io by my students:

- **Rot & Blau**
- **Animals vs Aliens**
- **Painting Punks**
- **Pixel Dancers**
- **Space Balls**
- **Alien Fight**
- **Tanques**
- **Running Pixel Girls**

There are also, some video games published on OpenProcessing by my students such as:

- **Navecitas v2.3**
- **BallMaze (Gyroscope-Phone)**
- **Play 2**
- **Game**
- **Cube**
- **Lineas Game**
- **Space Ship**
- **Pez.k**
- **Inversiones Semánticas**
- **Arkanoid**
- **TransmiKiller**
- **Esquivalo**
- **Car Obstáculos**
- **Rana**
- **Saltando Obstaculos**
- **Invaders 4 Real**
- **Come Bolitas**

In this respect, I also have taught video game subjects using the game engine Unity 3D such as: Virtual Reality and Artificial Intelligence for Video Games. Therefore, for this project I am using the experience and knowledge that I acquired with Unity 3D, to currently implement game mechanics in p5.js that allows build functional 2D video games. The general curriculum and the students results of these subjects using Unity 3D, can be seen in the following links:

- **Virtual Reality**
- **Artificial Intelligence for Video Games**

In addition, I studied the master's degree Cognitive Systems and Interactive Media in the Pompeu Fabra in Barcelona, Spain where I started developing interactive apps using Processing. In this master's I carried out the thesis *XIM expressions: Socializing with a non-anthropomorphic robot*. This thesis was elaborated using a full-body experiment developed in Unity 3D as can be seen in this link.

THEORY

The object-oriented programming paradigm allows to reuse code and to have the control of each object on the screen. It means, this paradigm allows to store the values of the properties of each object such as: position, size, color, velocity, acceleration, or availability status. These properties can be managed through methods that represent object behaviors such as: change color, change size, move up or move in any direction and bounce on the edges of the screen

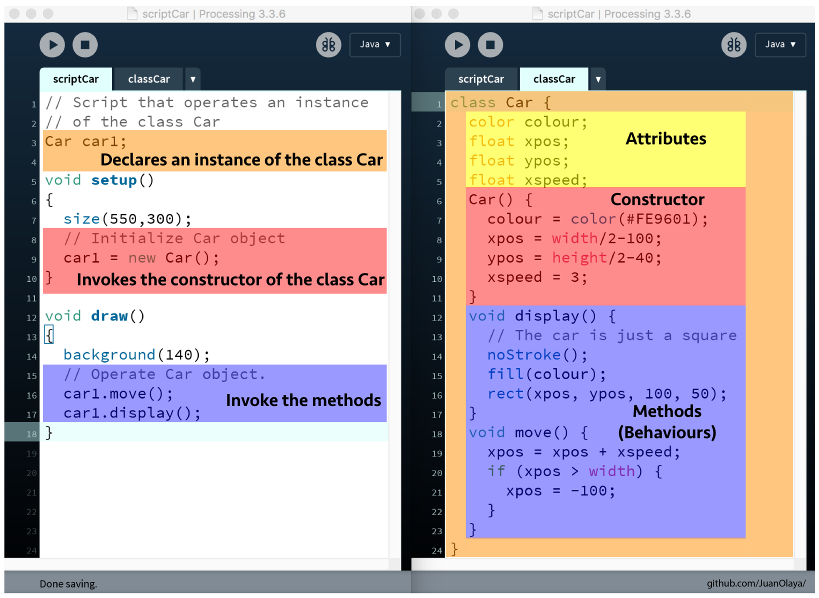
EXPECTED RESULTS *the project's expected results*

The result of this project is an online book that shows 20 lessons the code of the examples with explanations and proposal activities for teachers. The 20 lessons are listed below:

Lesson 1: One class, one object
Lesson 2: One class, two objects (and the constructor with arguments)
Lesson 3: One class, multiple objects (and how to store them on arrays)
Lesson 4: Using vectors
Lesson 5: Gravity
Lesson 6: Inheritance and Polymorphism
Lesson 7: Player movement with simultaneous keys
Lesson 8: Collisions
Lesson 9: Two Players Movement and Stop Player by Obstacle
Lesson 10: Jump
Lesson 11: Jump on platform
Lesson 12: Bullet
Lesson 13: Bullet with delay
Lesson 14: Screens
Lesson 15: Using sprites for animations
Lesson 16: Playing sounds
Lesson 17: Storing the score with JSON
Lesson 18: Sorting the ranking array
Lesson 19: More examples
Lesson 20: Publication on Itch.io

I described each lesson below:

Lesson 1: One class, one object
Based on the class car developed by Daniel Shiffman, I described in this section, the parts of a class such as: Constructor and Methods. I also described how to create an object in the class car, how to invoke the constructor and how to invoke its methods.
See the example sketch on OpenProcessing (p5.js)



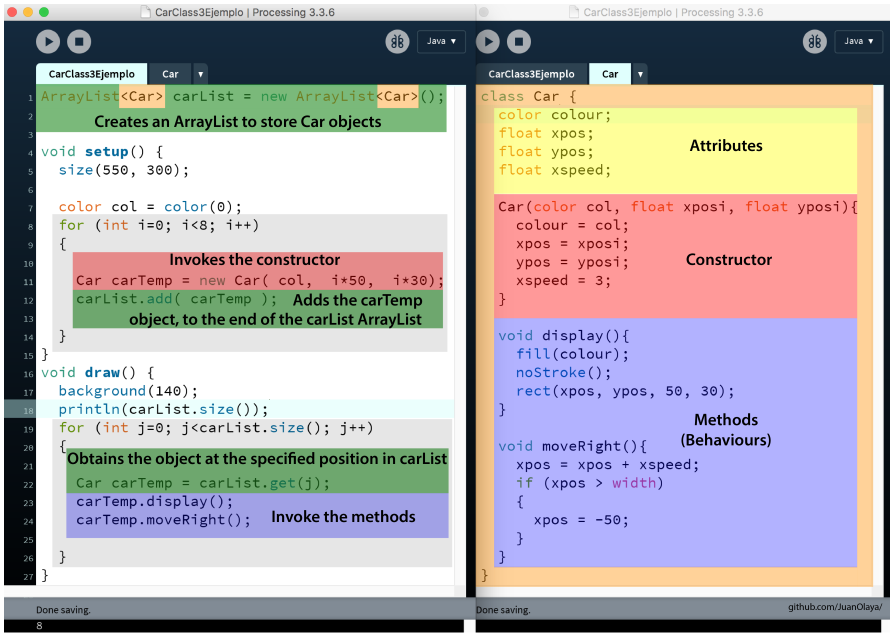
```
// Script that operates an instance
// of the class Car
Car car1;
void setup()
{
  size(550,300);
  // Initialize Car object
  car1 = new Car();
}

void draw()
{
  background(140);
  // Operate Car object.
  car1.move();
  car1.display();
}
```

```
class Car {
  color colour;
  float xpos;
  float ypos;
  float xspeed;
  Car() {
    colour = color(#FE9601);
    xpos = width/2-100;
    ypos = height/2-40;
    xspeed = 3;
  }
  void display() {
    // The car is just a square
    noStroke();
    fill(colour);
    rect(xpos, ypos, 100, 50);
  }
  void move() {
    xpos = xpos + xspeed;
    if (xpos > width) {
      xpos = -100;
    }
  }
}
```

Lesson 2: One class, two objects (and the constructor with arguments)

See the example sketch on OpenProcessing (p5.js)


CarClass3Ejemplo | Processing 3.3.6
CarClass3Ejemplo
Car
Creates an ArrayList to store Car objects
Invokes the constructor
Adds the carTemp object, to the end of the carList ArrayList
Obtains the object at the specified position in carList
Invoke the methods
Attributes
Constructor
Methods
(Behaviours)
Done saving.
8
github.com/JuanOlaya/

```
ArrayList<Car> carList = new ArrayList<Car>();

void setup() {
  size(550, 300);

  color col = color(0);
  for (int i=0; i<8; i++)
  {
    Car carTemp = new Car( col,  i*50,  i*30);
    carList.add( carTemp );
  }
}
void draw() {
  background(140);
  println(carList.size());
  for (int j=0; j<carList.size(); j++)
  {
    Car carTemp = carList.get(j);
    carTemp.display();
    carTemp.moveRight();

  }
}
```

```
class Car {
  color colour;
  float xpos;
  float ypos;
  float xspeed;

  Car(color col, float xposi, float yposi){
    colour = col;
    xpos = xposi;
    ypos = yposi;
    xspeed = 3;
  }

  void display(){
    fill(colour);
    noStroke();
    rect(xpos, ypos, 50, 30);
  }

  void moveRight(){
    xpos = xpos + xspeed;
    if (xpos > width)
    {
      xpos = -50;
    }
  }
}
```

Lesson 3: One class, multiple objects (and how to store them on arrays)

See the example sketch on OpenProcessing (p5.js)

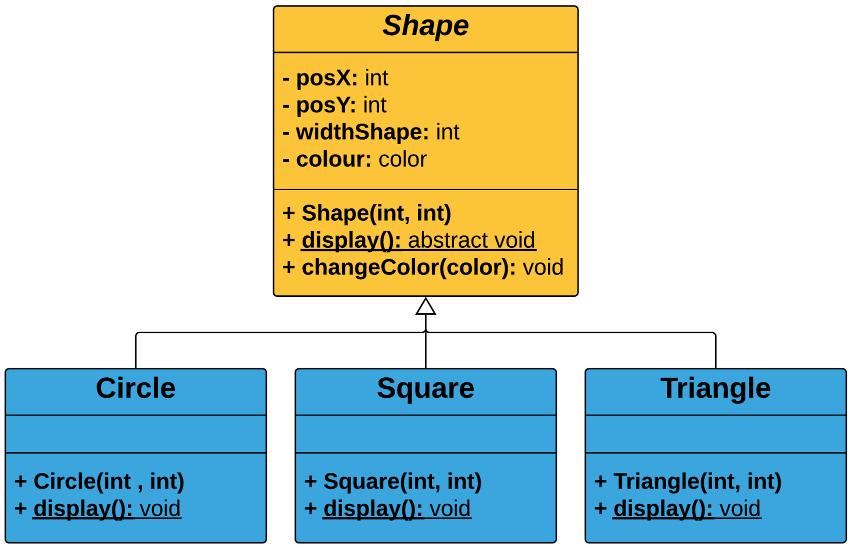
Lesson 4: Using vectors

See the example sketch on OpenProcessing (p5.js)

Lesson 5: Gravity + Border rebound

Implementation of class Ball
See the example sketch on OpenProcessing (Processing)

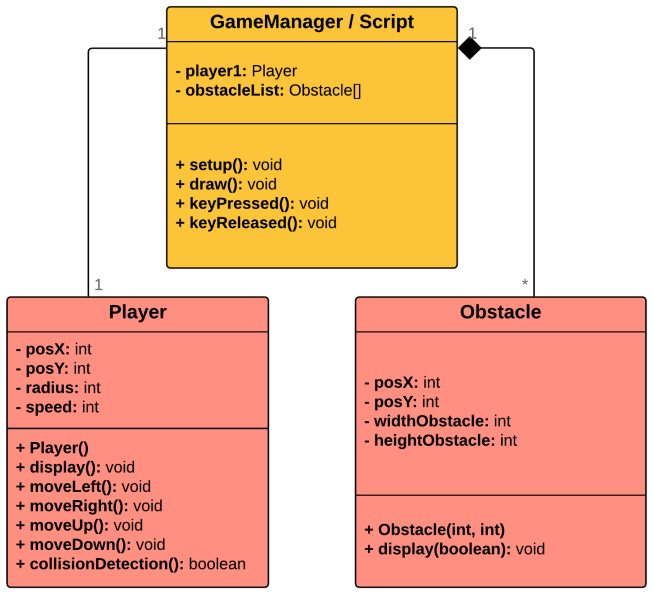
Lesson 6: Inheritance and Polymorphism
See the example sketch on OpenProcessing (Processing)



Lesson 7: Player movement
See the example sketch on OpenProcessing (p5.js)

Lesson 8: Collision Detection
This lesson shows the algorithm about how to detect collisions between a circle and a square. This method is based on Jeff Thompson's algorithm. See the example sketch on OpenProcessing (Processing)

In addition, this lesson presents how to communicate two classes to detect the collision between the player and the obstacles. See the example sketch on OpenProcessing (Processing)

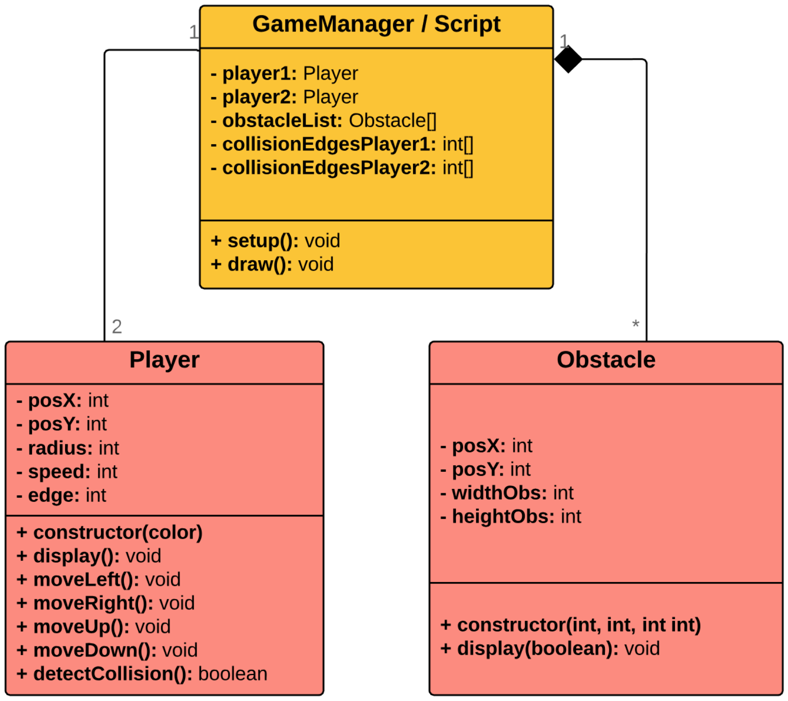


Lesson 9: Two Players movement and Stop Player by Obstacle

This lesson also presents how to communicate between two classes to detect the collision between the player and the obstacles and how to stop the player when it found an obstacle. See the example sketch on OpenProcessing (p5.js)

For this lesson I added to the detection collision algorithm an edge of the collision in order to be more accurate and be more realistic. The explanation examples can be seen in the links below:

- 8 edges collision detection. See the example sketch on OpenProcessing (Processing)
- 12 edges collision detection. See the example sketch on OpenProcessing (Processing)



Lesson 10: Jump
See the example sketch on OpenProcessing

Lesson 11: Jump on platform
See the example sketch on OpenProcessing (Processing)

Lesson 12: Bullet
See the example sketch on OpenProcessing (Processing)

Lesson 13: Bullet with delay
This lesson explains how to use the millis() function of p5.js to develop a delay between the fired bullets. See the example sketch about millis() function on OpenProcessing

Lesson 14: Screens and Buttons
See the example sketch on OpenProcessing

Lesson 15: Using sprites for animations
See the example sketch on OpenProcessing

Lesson 16: Playing sounds

Lesson 17: Storing the score with JSON

Lesson 18: Sorting the ranking array
This lesson explains how to use bubble sort algorithm to organize a list from highest to lowest by ranking

```
void bubbleSortAlgorithm() {
  for (int i=0; i<playersList.size()-1; i++) {
    for (int j=0; j<playersList.size()-1; j++) {
      if(playersList.get(j).score < playersList.get(j+1).score){
        Player playerAux1 = playersList.get(j);
        Player playerAux2 = playersList.get(j+1);
        playersList.set(j,playerAux2);
        playersList.set(j+1,playerAux1);
      }
    }
  }
}
```

Lesson 19: More examples
Maze: See the example sketch on OpenProcessing
and generate others behaviors to the objects such as: shake up-down or shake left-right depending on edge collision which will present in lesson 19.

Collision + Rebound + Shake: See the example sketch on OpenProcessing

Lesson 20: Publication on Itch.io
The activity for this lesson is create a user account on Itch.io and publish the video game on this platform. Is relevant upload a gameplay video. A list of some video games created by my students can be found on my itch.io profile in the link below:
https://juanolaya.itch.io/

EXPANDING POSSIBILITIES OF P5.JS how your project will expand the possibilities of Processing, either through software, community, or both

As seen in the inspirational video tutorials of Daniel Shiffman is possible to develop several 2D video games using Processing and p5.js. However, there are many game mechanics that have not yet been explored. In fact, there is not yet a central website or book that focus on the game development using Processing or p5.js.

Therefore, there is an opportunity to build and publish a curriculum that allow to teachers, students, educational institutions and enthusiasts, teach and learn how to develop 2D video games using p5.js.

In addition, is relevant to define a programming paradigm, as the object-oriented, as the paradigm to develop game mechanics and game managers in order to implement universal understandable and scalable solutions for game developers in training.

TARGET AUDIENCIES how your project speaks to the needs of a specific group, and how it will support the fostering of this community.

This online book is focused on the field of Computer Science Education for teachers and students who want to teach or learn the concepts about object-oriented programming using creative coding, exploring the concepts about the physics simulations and 2D game development.

This kind of teachers and students are found in conventional high schools and interdisciplinary bachelors or master's degrees about new media, electronic arts or human-computer interaction which have students with different backgrounds that must to learn the object-oriented programming paradigm to develop functional 2D video games with educational and entertainment approaches.

This project takes advantage of the student motivation and their meaningful experience playing video games, to invite them to create their own video games and turn into reality their video game ideas.

This curriculum focuses on advanced beginner students who have studied the basic topics of Processing or p5.js. These basic topics can be reviewed in the following featured books:

- Shiffman, D. (2009). ***Learning Processing****: a beginner's guide to programming images, animation, and interaction*. Morgan Kaufmann.

- McCarthy, L., Reas, C., & Fry, B. (2015). ***Getting Started with P5. js****: Making Interactive Graphics in JavaScript and Processing*. Maker Media, Inc.

FUTURE WORKS

- Develop a user sessions management for video games in p5.js
- Develop a multiplayer manager using WebSockets in p5.js
- Positioning of p5.js as a relevant 2D game engine