GRAPHICS SYSTEMS AND INTERACTION

Lesson 3

Abstract

Project “Basic Pong”
Creating multiple instances of the game
Project “Pong”

João Paulo Pereira
jjp@isep.ipp.pt

Table of Contents

Project “Basic Pong” 1

The coordinate system 2

To-do #1 – Create the net 2

To-do #2 – Create the rackets 3

To-do #3 – Set the rackets’ center positions 3

To-dos #4 and #5 – Add the rackets to the scene 3

To-do #6 – Update the rackets’ center positions 3

To-do #7 - Compute the rackets’ lower and upper boundaries 3

To-do #8 - Check the rackets’ lower and upper boundaries 3

To-do #9 – Compute the ball’s new center position 3

To-do #10 – Check if the ball hit the sidelines 4

To-dos #11 and #12 – Check if the ball hit the rackets 4

To-do #13 – Set the ball’s new center position 4

To-do #14 – Check if a player scored 4

To-do #15 – Check if the game is over 4

Creating multiple instances of the game 5

Project “Pong” 6

To-do at home #1 – Allow players to move their rackets back and forth 6

To-do at home #2 – Let players control the ball’s speed, direction, and spin 7

To-do at home #3 – Speed attenuation over time 7

To-do at home #4 – Spin attenuation over time 7

To-do at home #5 – Changing the racket’s size and speed 7

References 9

Table of Figures

Figure 1 – Project “Basic Pong” 1
Figure 2 – The coordinate system 2
Figure 3 – Updating the ball position 3
Figure 4 – Checking if the ball hit the sidelines and computing the ball's new direction 4
Figure 5 – Checking if the ball hit the rackets and computing the ball's new direction 4
Figure 6 – Multiple instances of the game 5
Figure 7 – Project “Pong” 6

Table of Equations

Equation 1 – Parametric form of the circle equation 3
Equation 2 – Speed attenuation over time 7
Equation 3 – Spin attenuation over time 7

Project “Basic Pong”

The aim of this three.js [1] project is to recreate the famous arcade video game Pong (Figure 1) [2].

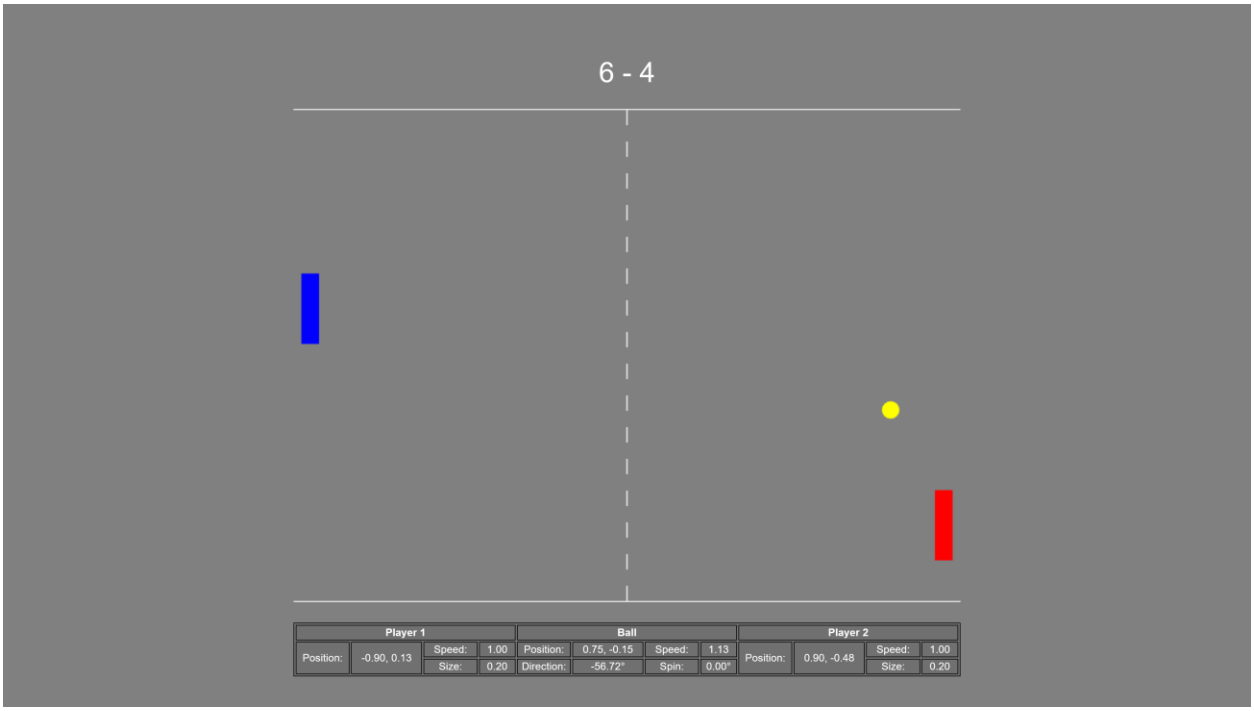

Figure 1 – Project “Basic Pong”

Download the folder “Basic_Pong_template”. The project is composed by eight files:

- “Pong.html” and “Multiple_Pongs.html”
- “default_data.js”
- “merge.js”
- “pong_template.js”
- “table_template.js”
- “player_template.js”
- “ball_template.js”

The scene, camera and renderer are created in the first two files, along with the instantiation of the game(s), and the definition of an animation function and a callback that processes window resizing events.

The class Pong is declared in file “pong_template.js”. It includes the constructor, which creates the game (composed by a table, two players and a ball), the score and the status display areas, callbacks to process blur/focus and keyboard events, and the update method, responsible for regularly update the game state, check whether a point has been scored, whether the game is over, etc.

The table (two continuous sidelines and a dashed line representing the net) is modeled in class Table (file “table_template.js”).

The class Player is defined in file “player_template.js” and models both players rackets (colored rectangles). It also updates racket positions and checks whether their lower and upper limits have been reached.

Finally, the ball is modeled in class Ball, which is described in file “ball_template.js”. The class includes a method to verify whether the ball has hit the sidelines or rackets and update its position and direction accordingly.

Default data such as colors, dimensions, speeds, players keys, etc. are defined in file “default_data.js”. They can be redefined in file “Pong_template.html” when instantiating the game.

The function merge is defined in file “merge.js”. It performs shallow / deep object merging and is responsible for merging default and post-defined data.

Your assignment is to write the code that handles some of the game’s functions, namely: model the net and the rackets; set the rackets’ positions and add these objects to the scene; update the rackets’ positions; compute and check their lower and upper boundaries; update the ball position; check if the ball hit the sidelines or the rackets and compute the ball’s new direction accordingly; finally, check if a player scored and if the game is over.

The coordinate system

Assume that the 3D Cartesian coordinate system is identical to the one specified in project “Watch” (Figure 2). Again, since we are dealing with a 2D design, the Z-axis and Z-coordinates can be disregarded most of the time.

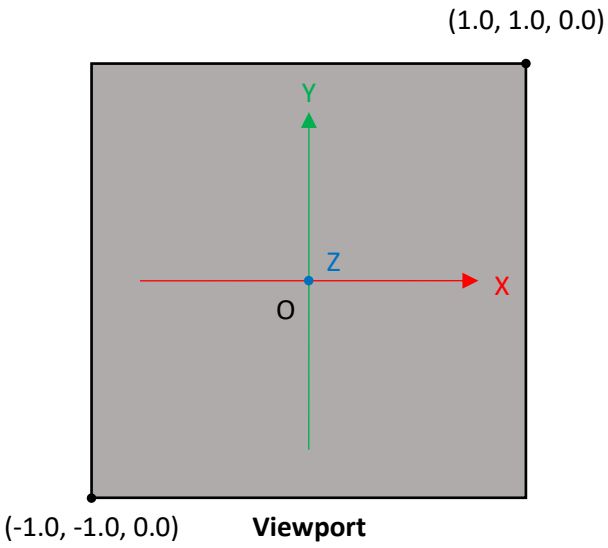

Figure 2 – The coordinate system

To-do #1 – Create the net

Open the file “table_template.js” and look for comment “To-do #1”. Follow the instructions.

Code example, class and method to consider: Dashed Lines Example [3], LineDashedMaterial [4] and Line.computeLineDistances [5].

To-do #2 – Create the rackets

Open the file "player_template.js" and look for comment "To-do #2". Follow the instructions.

Code example and class to consider: PlaneGeometry [6].

To-do #3 – Set the rackets' center positions

Look for comment "To-do #3" and follow the instructions.

To-dos #4 and #5 – Add the rackets to the scene

Open the file "pong_template.js" and look for comments "To-do #4" and "To-do #5". Follow the instructions. You should now see both rackets.

Start/pause the game by pressing the "Space" key. Resume/restart it whenever needed.

To-do #6 – Update the rackets' center positions

Open the file "player_template.js" and look for comment "To-do #6". Follow the instructions.

To-do #7 - Compute the rackets' lower and upper boundaries

Look for comment "To-do #7" and follow the instructions.

To-do #8 - Check the rackets' lower and upper boundaries

Look for comment "To-do #8" and follow the instructions.

To-do #9 – Compute the ball's new center position

Use the parametric form of the circle equation to compute the ball's new center position (Figure 3 and Equation 1) [7].

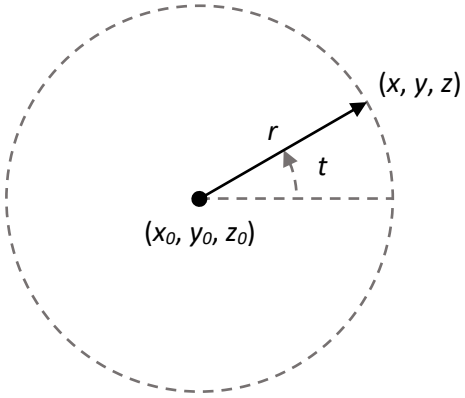

Figure 3 – Updating the ball position

$$\begin{cases} x = r * \cos(t) + x_0 \\ y = r * \sin(t) + y_0 \\ z = z_0 \end{cases}$$

Equation 1 – Parametric form of the circle equation

Where:

- (x, y, z) are the ball's new center coordinates

- (x_0, y_0, z_0) are the ball's current center coordinates
- r = (*ball speed* * *elapsed time*) is the distance covered by the ball
- t is the ball direction (expressed in radians)

Open the file "ball_template.js" and look for comment "To-do #9". Follow the instructions.

To-do #10 – Check if the ball hit the sidelines

Carefully examine Figure 4 and try to figure out the new direction of the ball. It depends only on the current direction and is identical for the rebounds on both the lower and upper sidelines.

Look for comment "To-do #10" and follow the instructions.

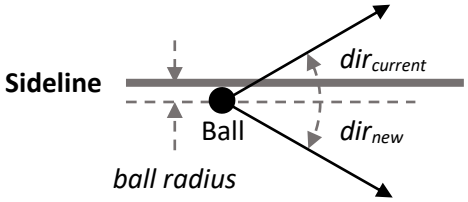

Figure 4 – Checking if the ball hit the sidelines and computing the ball's new direction

To-dos #11 and #12 – Check if the ball hit the rackets

Look at Figure 5 and try figuring out the ball's new direction. It depends solely on the current direction and is the same for rebounds on both players rackets.

Look for comments "To-do #11" and "To-do #12", and follow the instructions.

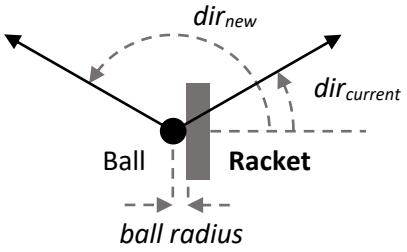

Figure 5 – Checking if the ball hit the rackets and computing the ball's new direction

To-do #13 – Set the ball's new center position

Look for comment "To-do #13" and follow the instructions.

To-do #14 – Check if a player scored

A player scores when the ball passes the end of the opposite side of the table.

Look for comment "To-do #14" and follow the instructions.

To-do #15 – Check if the game is over

The game ends when a player's score reaches a given threshold.

Look for comment "To-do #15" and follow the instructions.

Creating multiple instances of the game

You can create multiple instances of the game (as in projects “Basic Watch” and “Interactive Watch”) just by changing the contents of file “Pong.html”. Nevertheless, it won’t be easy to play under these conditions (Figure 6).

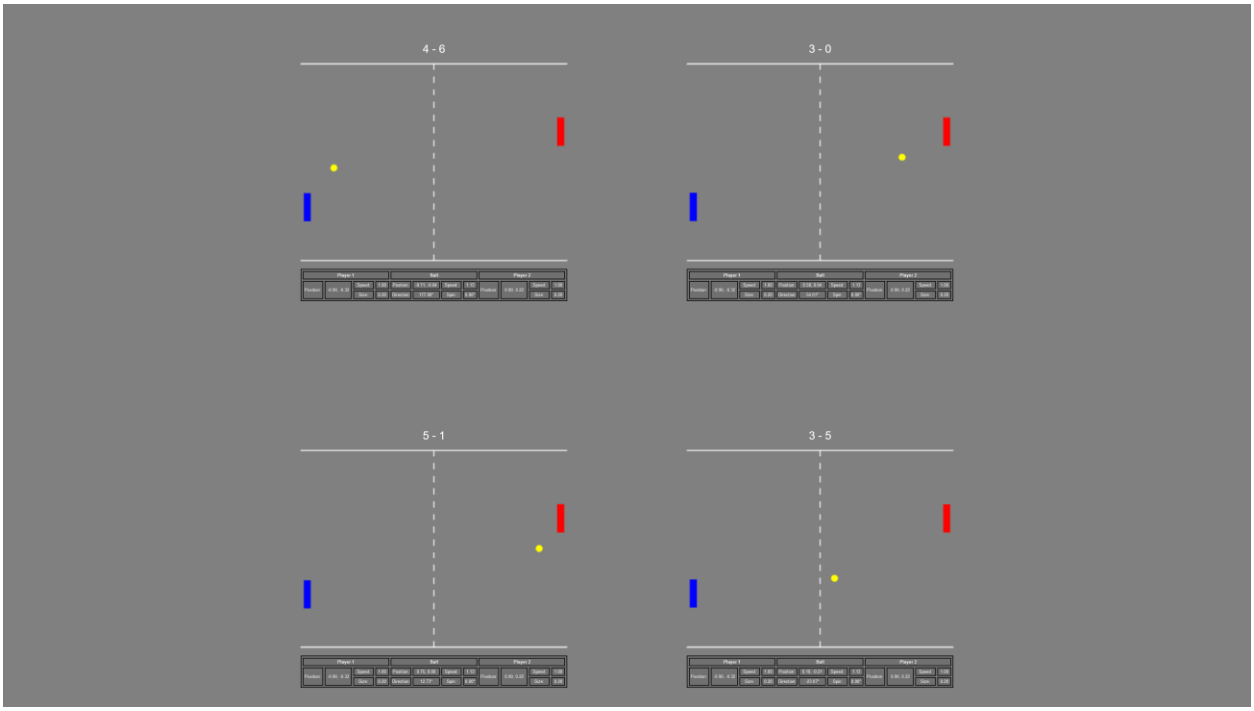

Figure 6 – Multiple instances of the game

Project “Pong”

This is a refinement of project “Basic Pong”. Differences are as follows ():

- Players can also move their rackets back and forth
- Players now control:
  - The ball speed
  - The ball direction
  - The ball spin (Magnus effect) [8]
- The ball's speed and spin are attenuated over time
- Whenever a player scores, the corresponding racket becomes smaller and faster
- Whenever a player concedes, the corresponding racket becomes larger and slower

Your assignment is to adapt project “Basic Pong” to accommodate these refinements.

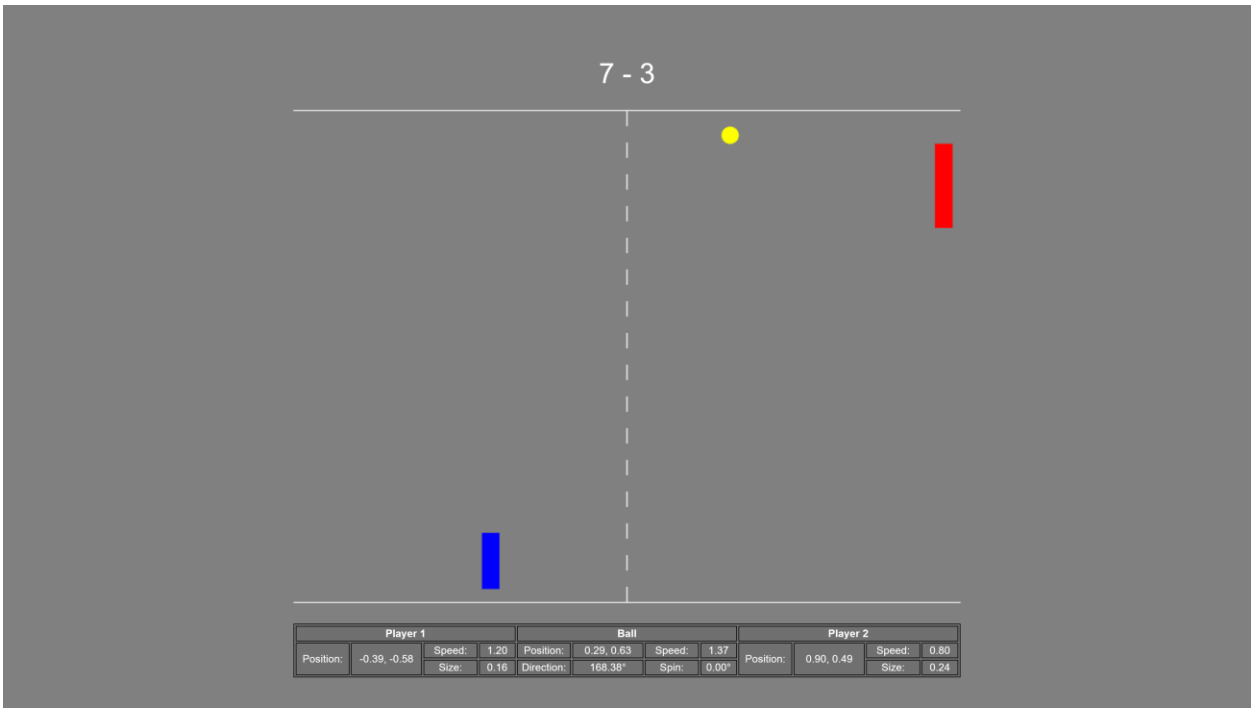

Figure 7 – Project “Pong”

To-do at home #1 – Allow players to move their rackets back and forth

You must define four additional keys. For example:

- Player 1: “A” (backward) and “D” (forward)
- Player 2: “L” (backward) and “J” (forward)

You have to define two additional limits for the rackets’ center positions: the rear and front boundaries. For example:

- Rear boundary: ±95% of the table’s half X-dimension (+ or - depending on the racket being considered)

- Front boundary: ±15% of the table's half X-dimension (+ or - depending on the racket being considered)

Finally, you must create two additional methods: checkRearBoundary and checkFrontBoundary, which should be called whenever players move their rackets back or forth.

To-do at home #2 – Let players control the ball's speed, direction, and spin

Whenever the ball hits a racket consider changing the following parameters:

- Its speed may depend on whether the racket is moving forward (faster) or backward (slower)
- Its direction may be a function of the hit point Y-coordinate in relation to the racket's center Y-coordinate
- Its spin (an angle) may vary depending on whether the racket is moving up or down (for example, *spin* = ±1° depending on the racket being considered and the direction it is moving; add the spin to the ball direction).

To-do at home #3 – Speed attenuation over time

The ball speed may be a function of the elapsed time. For example:

$$speed = speed_0 * (1.0 + speedAttenuation)^{elapsedTime}$$

Equation 2 – Speed attenuation over time

Where:

- *speed* is the ball's new speed
- $speed_0$ is the ball's current speed
- *speedAttenuation* is the speed attenuation (e.g., -20% per second)
- *elapsedTime* is the elapsed time in seconds

To-do at home #4 – Spin attenuation over time

The ball spin may also be a function of the elapsed time. For example:

$$spin = spin_0 * (1.0 + spinAttenuation)^{elapsedTime}$$

Equation 3 – Spin attenuation over time

Where:

- *spin* is the ball's new spin
- $spin_0$ is the ball's current spin
- *spinAttenuation* is the spin attenuation (e.g., -50% per second)
- *elapsedTime* is the elapsed time in seconds

To-do at home #5 – Changing the racket's size and speed

Whenever a player scores, the corresponding racket may become smaller and faster; and the opposite player's racket, on the contrary, may become larger and slower. Note, however, that when increasing the size of a racket, you must ensure that it won't exceed the table's lower and upper boundaries.

References

[1] Three.js, "Three.js – JavaScript 3D Libray," [Online]. Available: https://threejs.org. [Accessed 25 July 2021].

[2] Wikipedia, "Pong," [Online]. Available: https://en.wikipedia.org/wiki/Pong. [Accessed 05 August 2021].

[3] Three.js, "three.js – dashed lines example," [Online]. Available: https://threejs.org/examples/webgl_lines_dashed.html. [Accessed 05 August 2021].

[4] Three.js, "LineDashedMaterial," [Online]. Available: https://threejs.org/docs/api/en/materials/LineDashedMaterial.html. [Accessed 05 August 2021].

[5] Three.js, "Line.computeLineDistances," [Online]. Available: https://threejs.org/docs/?q=line#api/en/objects/Line.computeLineDistances. [Accessed 05 August 2021].

[6] Three.js, "PlaneGeometry," [Online]. Available: https://threejs.org/docs/api/en/geometries/PlaneGeometry.html. [Accessed 05 August 2021].

[7] Wikipedia, "Circle," [Online]. Available: https://en.wikipedia.org/wiki/Circle. [Accessed 25 July 2021].

[8] Wikipedia, "Magnus effect," [Online]. Available: https://en.wikipedia.org/wiki/Magnus_effect. [Accessed 05 August 2021].