GRAPHICS SYSTEMS AND INTERACTION

Lesson 3

Abstract

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

João Paulo Pereira
jjp@isep.ipp.pt

Table of Contents

Table of Figures

Table of Equations

Project “Basic Pong”

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

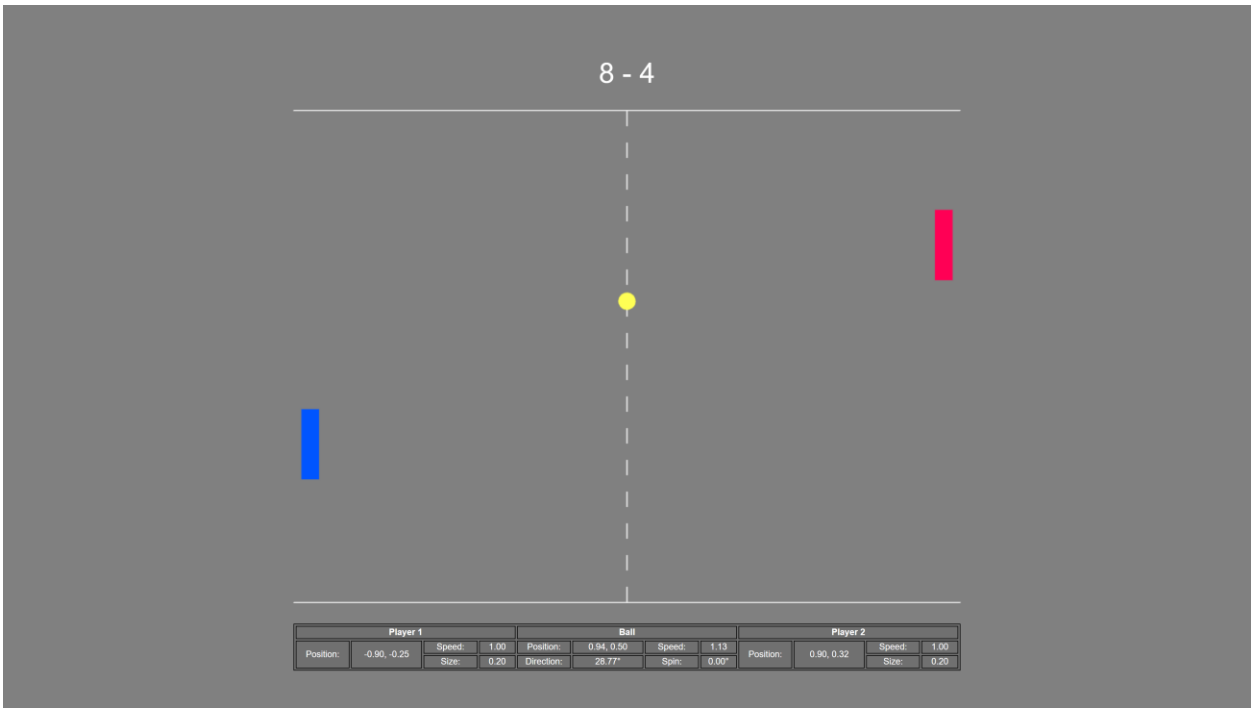

Figure 1 – Project “Basic Pong”

Download the folder “Basic_Pong_template”. The project is composed by two subfolders and nine files:

- “clips/”
- “textures/”
- “Pong.html” and “Multiple_Pongs.html”
- “default_data.js”
- “merge.js”
- “audio.js”
- “pong_template.js”
- “table_template.js”
- “player_template.js”
- “ball_template.js”

The scene, camera and renderer are created in the two HTML 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 if a point has been scored, if 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 if 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 if the ball has hit the sidelines or rackets and update its position and direction.

Default data such as colors, dimensions, speeds, players keys, etc. is defined in file "default_data.js". It can be redefined in file "Pong_template.html" when instantiating the game.

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

The class Audio is defined in file "audio.js" and is responsible for loading, configuring, and playing the audio clips stored in folder "clips" (project "Sketched Pong" only).

Folder "textures" contains texture image files (project "Sketched Pong" only).

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; 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 projects "Basic Watch" and "Interactive 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.

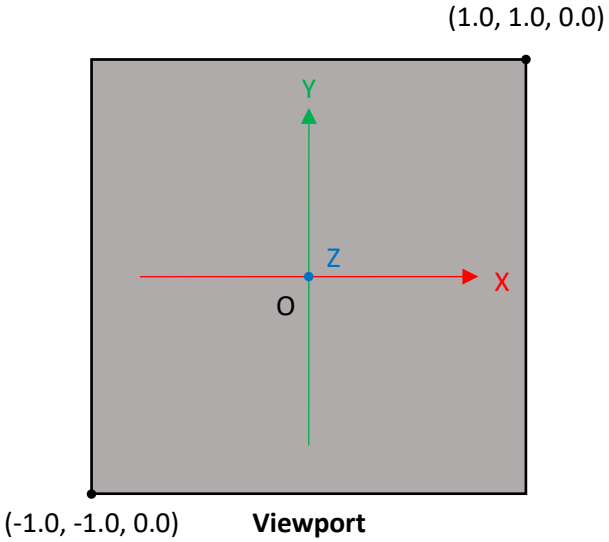

Figure 2 – The coordinate system

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].

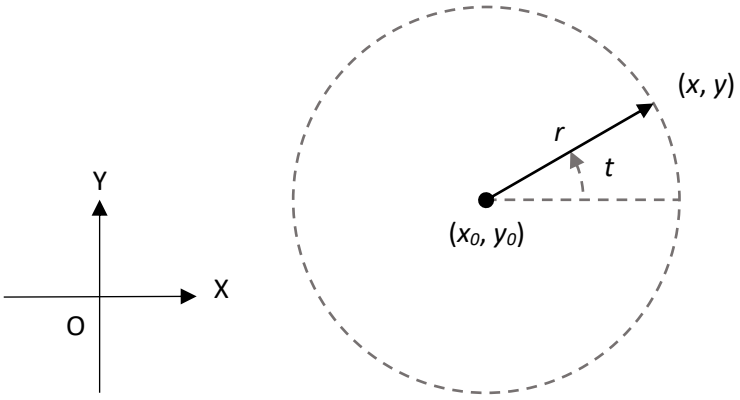

Figure 3 – Updating the ball position

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

Equation 1 – Parametric form of the circle equation

Where:

- (x, y) are the ball's new center coordinates
- (x_0, y_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.

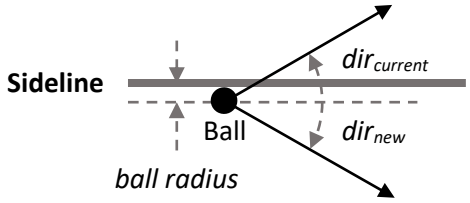

Figure 4 – Checking if the ball hit the sidelines and computing the ball's new direction

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". Follow the instructions.

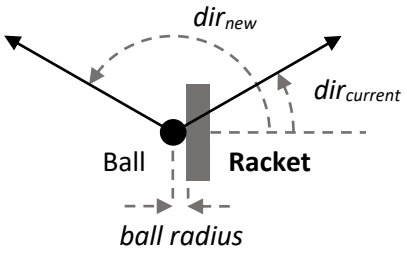

Figure 5 – Checking if the ball hit the rackets and computing the ball's new direction

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.

Open the file “pong_template.js” and look for comment “To-do #14”. 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).

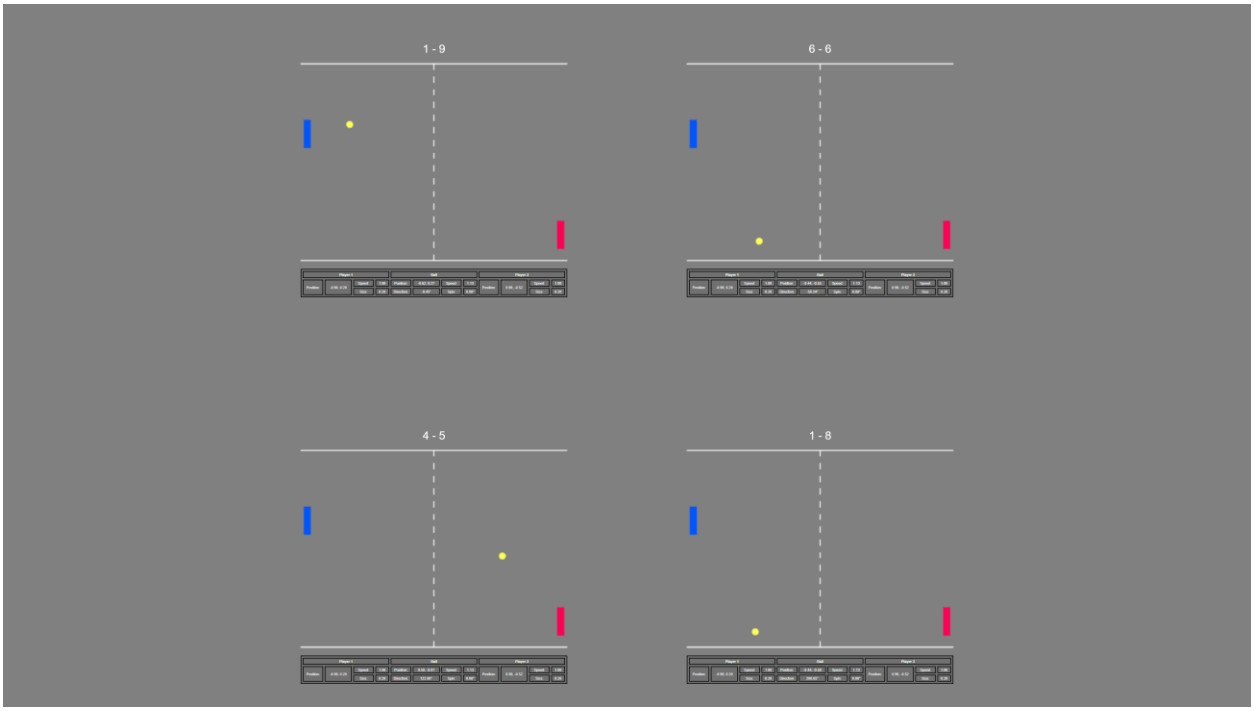

Figure 6 – Multiple instances of the game

Project “Pong”

This is a refinement of project “Basic Pong”. Differences are as follows (Figure 7):

- 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.

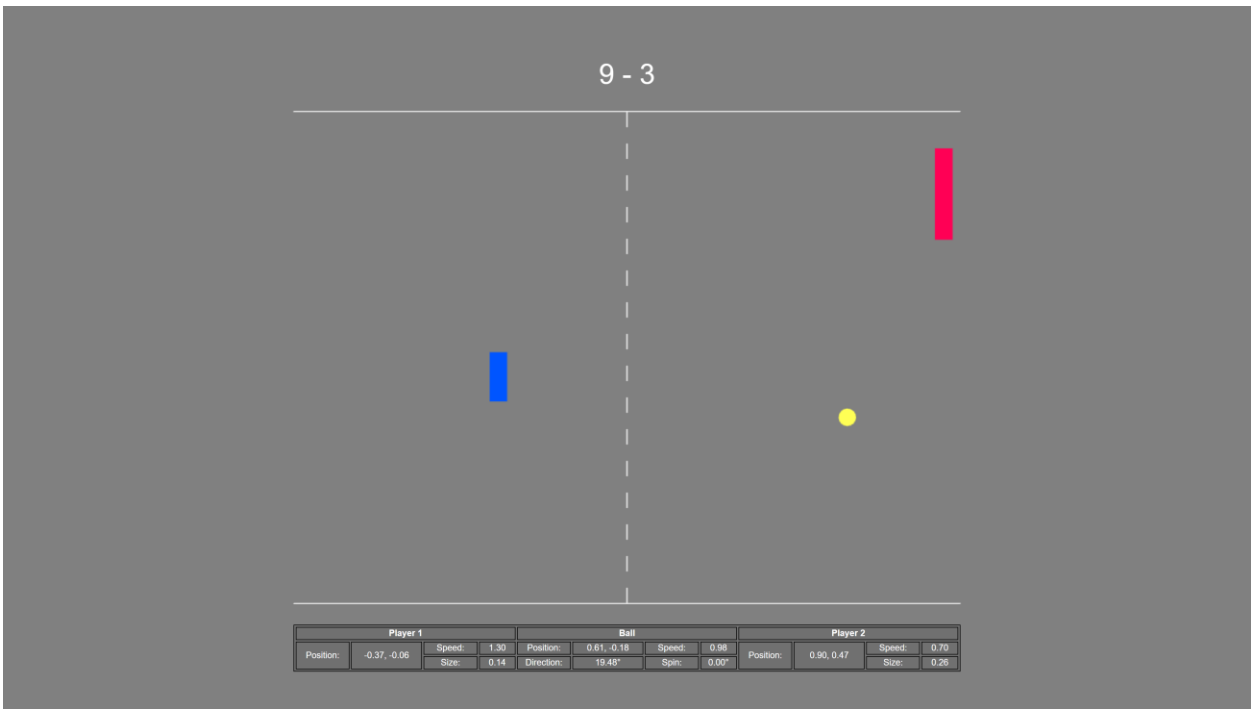

Figure 7 – Project “Pong”

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.

Project “Sketched Pong”

Let’s grab a pencil and sketch Pong on a crumpled sheet of paper. Major differences from project “Pong” should be as follows (Figure 8):

- Textures are applied to the background, ball and rackets
- Rackets and ball sizes are slightly greater
- The font family used to display information is different
- There’s an initial overlay with a starting button. Users can either click on the button or press the “Enter” key / “Space” bar to start the game
- Three different sound effects were added to the game: whenever the ball hits the sidelines or rackets, and each time a player scores
- Users can enable/disable sound effects by pressing “Shift” together with the predefined audio enable/disable key
- Whenever users enable/disable sound effects, a small overlay provides the corresponding feedback

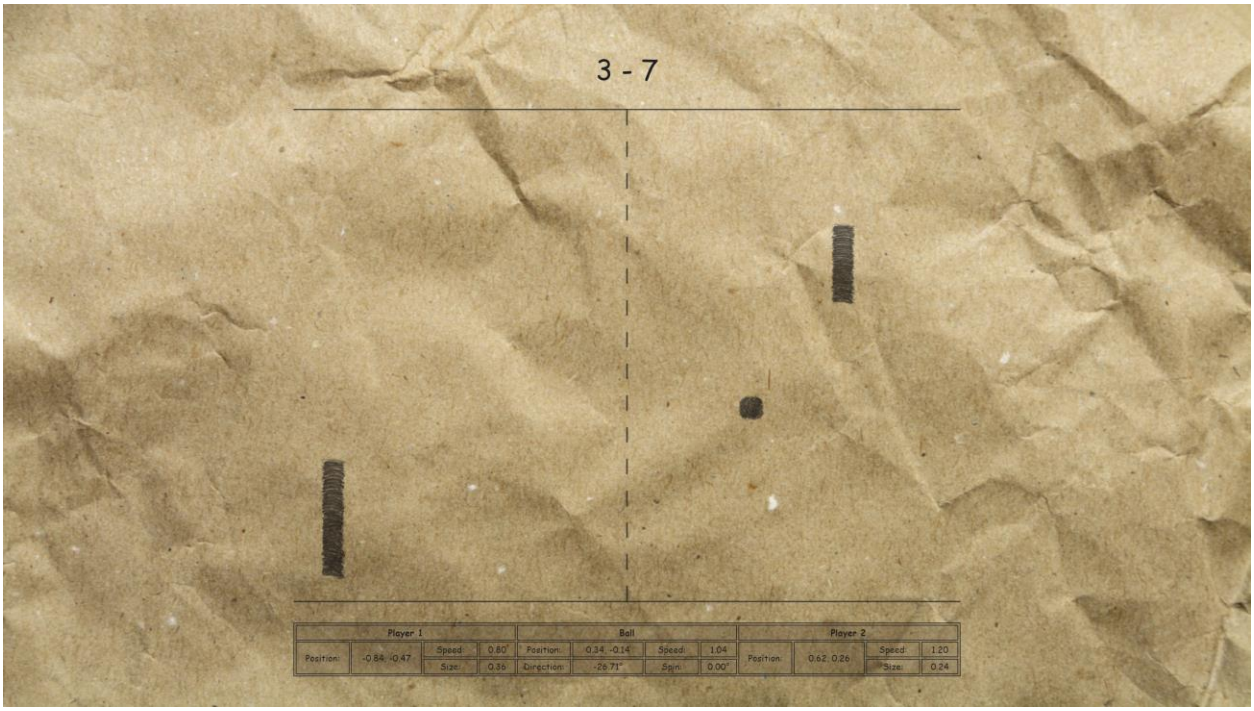

Figure 8 – Project “Sketched Pong”

To-do at home #6 – Apply textures to objects

Loading, configuring, and applying textures to the scene background and to objects materials can be an easy task. Consider the following code examples and classes: TextureLoader [9], Texture [10]. Then try assigning:

- Image “textures/crumpled-paperboard.jpg” to the scene background
- Image “textures/pencil-scribble-texture-1.png” to the ball material
- Image “textures/pencil-scribble-texture-3.png” to rackets’ material

You should set each texture color space to THREE.SRGBColorSpace [11].

You might also consider configuring magnification and minification filters for the ball and rackets textures as THREE.NearestFilter. Default values (THREE.LinearFilter and THREE.LinearMipmapLinearFilter, respectively) usually provide better results, but in this particular project the pencil stroke will probably look sharper if you choose THREE.NearestFilter. Available values are as follows:

Magnification filters

- THREE.NearestFilter
- THREE.LinearFilter

Minification filters

- THREE.NearestFilter
- THREE.NearestMipmapNearestFilter
- THREE.NearestMipmapLinearFilter
- THREE.LinearFilter
- THREE.LinearMipmapNearestFilter
- THREE.LinearMipmapLinearFilter

To-do at home #7 and #8 – Increase rackets initial size and the ball radius

Slightly increase rackets initial size. You could accomplish this by changing the appropriate default value in file "default_data.js", but the idea consists of leaving the contents of such file unaltered and editing the file "Pong.html" instead. Open the latter and look for the line containing the comment "Player 1 parameters".

The comment is preceded by some parameters within braces. Their role is to override the corresponding default values in file "default_data.js". Try adding the following parameter:

```
size: new THREE.Vector2(0.075, 0.3)
```

The default value will be disregarded. Do the same for Player 2.

Now slightly increase the ball radius. Look for the line with the comment "Ball parameters". Add the following parameter within the existing braces:

```
radius: 0.0375
```

The default value will be superseded.

To-do at home #9 – Change the font family

Open the file "Pong.html" and change the font family (look for "font-family"). For example, replace:

```
font-family: Arial, sans-serif;
```

by:

```
font-family: "Comic Sans MS", cursive;
```

“Comic Sans MS” is an example of a font family name, while “cursive” is the name of a generic family. You should always include at least one generic family name in a font-family list, since there’s no guarantee that any given font is available. This lets the browser select an acceptable fallback font when necessary [12].

To-do at home #10 – Create a startup overlay containing a “Start” button

Create a startup overlay with a “Start” button (Figure 9). Players should click on the button or press the “Enter” key. The overlay then disappears, exposing the game screen.

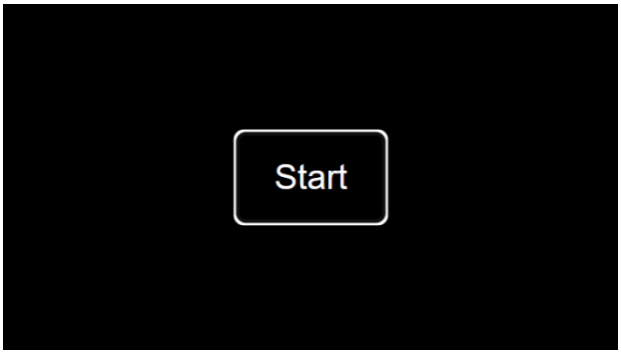

Figure 9 – Startup overlay

Go to project folder “Cube – NT” (downloaded from Moodle when installing Three.js) and open the file “Cube.html” in your code editor. Try to identify and understand the code snippets related to this task and insert them into the appropriate places of the file “Pong.html”. You must:

- Configure the overlay style (look for keyword "#overlay")
- Configure the overlay button style (look for "#overlay button")
- Create two HTML elements: a element (the overlay) containing a <button> element (the “Start” button). Look for “id="overlay"” and “id="start-button"”
- Get a reference to the button element and assign it to constant “startButton” (look for “document.getElementById("start-button")”)
- Register the event handler to be called when the user clicks on the button. The handler should be a function named “initialize” (look for “startButton.addEventListener("click", initialize)”)
- Set the focus on the button, so that it will receive keyboard and similar events (such as pressing the “Enter” key) by default (look for “startButton.focus()”)
- In function initialize:
    - Remove the event handler to be called when the user clicks on the button (look for “startButton.removeEventListener("click", initialize)”)
    - Remove the startup overlay (look for “document.getElementById("overlay").remove()”)

To-do at home #11 – Add sound effects to the game

Loading, configuring, and playing sounds can be easy too. Audio objects can be non-positional (global) or positional. Consider these code examples and classes: AudioListener [13], AudioLoader [14], Audio [15], PositionalAudio [16]. Then do the following:

- Create an audio listener
- Create an audio buffer loader

- Create three non-positional (global) audio sources and associate them to the audio listener:
  - Clip “clips/ping_pong_8bit_plop.ogg”
  - Clip “clips/ping_pong_8bit_beeep.ogg”
  - Clip “clips/ping_pong_8bit_peeeeeep.ogg”
- Add the audio listener to the camera
- Play the first clip (“plop”) every time the ball hits the sidelines
- Play the second clip (“beeep”) whenever the ball hits the rackets
- Play the third clip (“peeeeeep”) each time a player scores

To-do at home #12 – Define a key for enabling/disabling sound effects

You should define an extra key (or key combination) to allow players to turn audio on and off. Also consider adding key information to the help table at the bottom of the screen (Figure 10).

Open the file “pong_template.js” and look for comment “Display players keys” in function update. Change the contents of variable “text”. Get useful information about the HTML <table> element in the following web pages: HTML tables [17], <table> The Table element [18], HTML Tables [19] and HTML table tag [20]

Figure 10 – Enable/disable sound effects (help information)

To-do at home #13 – Add feedback about the current state of sound effects

Players should get some visual feedback every time they turn audio on and off (Figure 11).

Figure 11 – Enable/disable sound effects (current state feedback)

Open the file “pong_template.js” and proceed as follows:

- Create an HTML element (a <table>, for example) and configure its style
- Change the element’s visibility to “hidden”
- Append the element as a child of existing HTML element “container” (referenced in the code by a constant with the same name)
- Each time a player presses the audio enable/disable key:
  - Change the contents of the element to “**Audio:** enabled” or to “**Audio:** disabled” as appropriate
  - Change the element’s visibility to “visible”
  - Set a timer and specify the amount of time (for example, 1000 milliseconds) it should wait before a given function or piece of code is executed [21]
  - Once the timer expires, set the element’s visibility back to “hidden”

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].

[9] Three.js, "TextureLoader," [Online]. Available: https://threejs.org/docs/api/en/loaders/TextureLoader.html. [Accessed 08 March 2024].

[10] Three.js, "Texture," [Online]. Available: https://threejs.org/docs/api/en/textures/Texture.html. [Accessed 27 October 2021].

[11] Three.js, "Color management," [Online]. Available: https://threejs.org/docs/manual/en/introduction/Color-management.html. [Accessed 08 March 2024].

[12] Mozilla, "font-family," [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/CSS/font-family. [Accessed 05 August 2024].

[13] Three.js, "AudioListener," [Online]. Available: https://threejs.org/docs/api/en/audio/AudioListener.html. [Accessed 06 August 2024].

[14] Three.js, "AudioLoader," [Online]. Available: https://threejs.org/docs/api/en/loaders/AudioLoader.html. [Accessed 06 August 2024].

[15] Three.js, "Audio," [Online]. Available: https://threejs.org/docs/api/en/audio/Audio.html. [Accessed 06 August 2024].

[16] Three.js, "PositionalAudio," [Online]. Available: https://threejs.org/docs/api/en/audio/PositionalAudio.html. [Accessed 06 August 2024].

[17] Mozilla, "HTML tables," [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/HTML/Tables. [Accessed 08 August 2024].

[18] Mozilla, "<table>: The Table element," [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/table. [Accessed 08 August 2024].

[19] W3Schools, "HTML Tables," [Online]. Available: https://www.w3schools.com/html/html_tables.asp. [Accessed 08 August 2024].

[20] W3Schools, "HTML table tag," [Online]. Available: https://www.w3schools.com/tags/tag_table.asp. [Accessed 08 August 2024].

[21] Mozilla, "setTimeout() global function," [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/setTimeout. [Accessed 08 August 2024].