# **Lesson 3:** Losing Health and UI

## **Objective:** Program a Lose condition to the Player and display its Health through a User Interface Bar.

**Time:** 30 Minutes

**Level:** 2 - Beginner

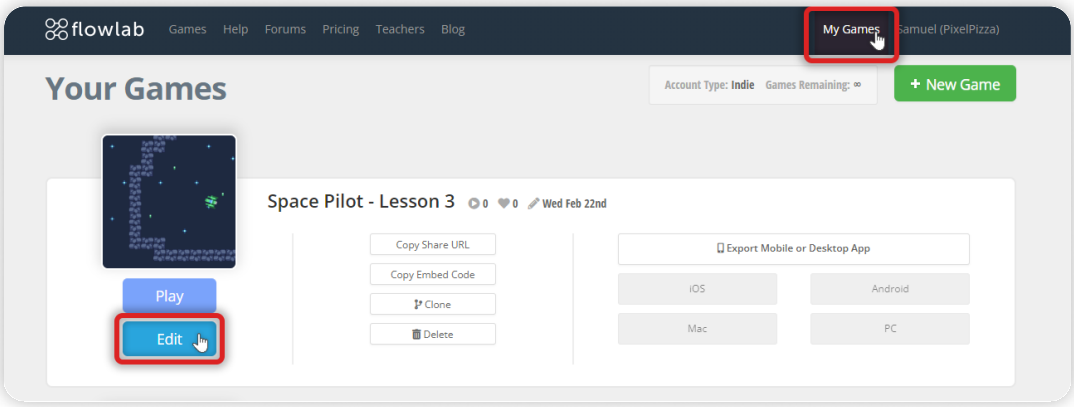
**Description:** Basic introduction to Physics drag and the User Interface layer. Using custom logic, program a Health Bar for the Player.

## Step 1

**Edit your Game**

Log in and start at your “My Games” page https://flowlab.io/game/list

Then, click “Edit” next to your game to open the game editor.



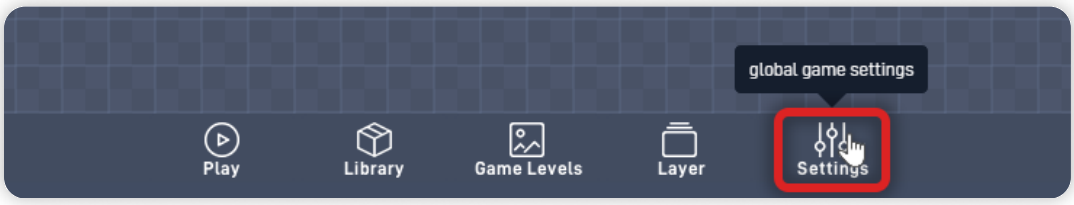

## Step 2

**Add physics Drag & adjust the player's Speed**

In the game's current state, after moving the Player spaceship, it drifts away in space when we are not applying a physical force - or in other words, when we are not pressing the Up key to move the ship forward.

This drift makes the game controls feel unresponsive and complicated, which can lead to a frustrating gameplay experience.
To adjust this “drift’s” strength, we need to change the **Physics Drag** - *A global physics value that removes velocity from all moving objects in each frame.*

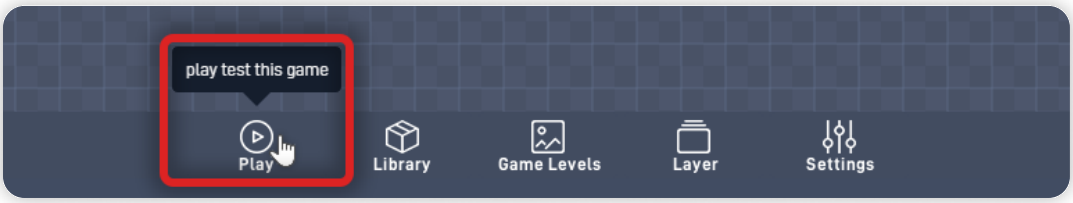
Set the Physics Drag by clicking on the Settings button on the bottom toolbar.



Click on “Advanced”, and then change the Physics Drag from “0.2” to “20”.
Click “OK” on the Settings panel to save your changes.



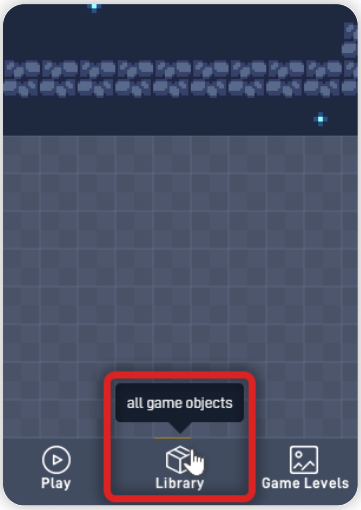
Now, click the “Play” button on the bottom toolbar and see how the Physics Drag value affects your game.

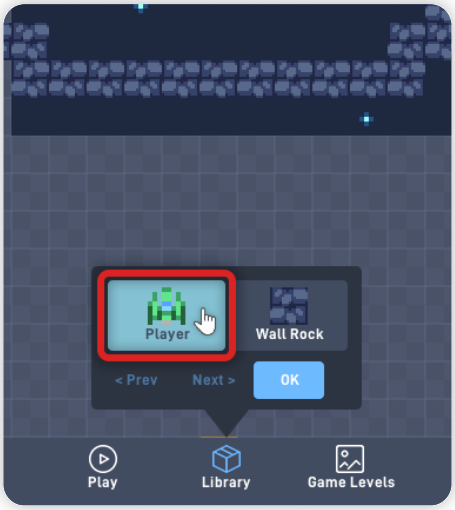


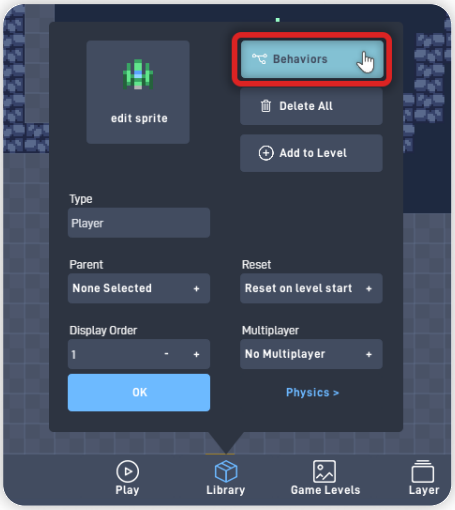

*You can now see that, the Player Ship moves really slowly, the controls feel stiffer, but the spaceship doesn’t float away in space forever anymore.*
*In this case, it’s precisely what we want; Now, let’s adjust the “Player” object movement speed to make it move faster.*

Press ESC on your keyboard to switch from the Play mode to the Editor.

Click on the Library button on the bottom toolbar to open the Object Library.
Click on the “Player” to open the object properties panel, and then click “Behaviors” to open the Behavior editor.

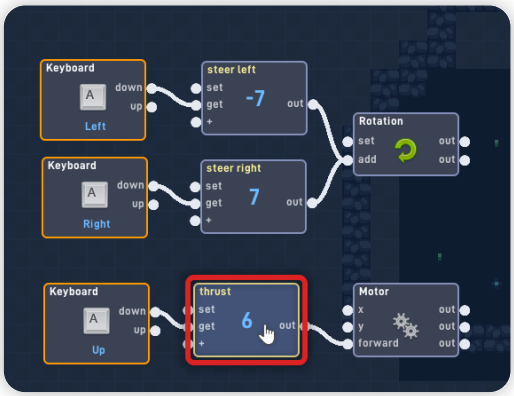


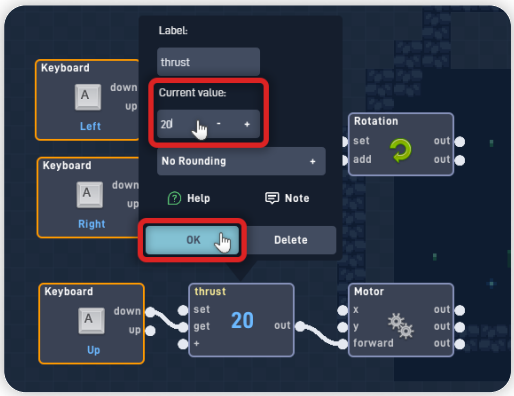



Inside the “Player” Behavior editor, double-click on the “Ship Controls” bundle to open its content and move the behaviors out of view so you can see the game below.

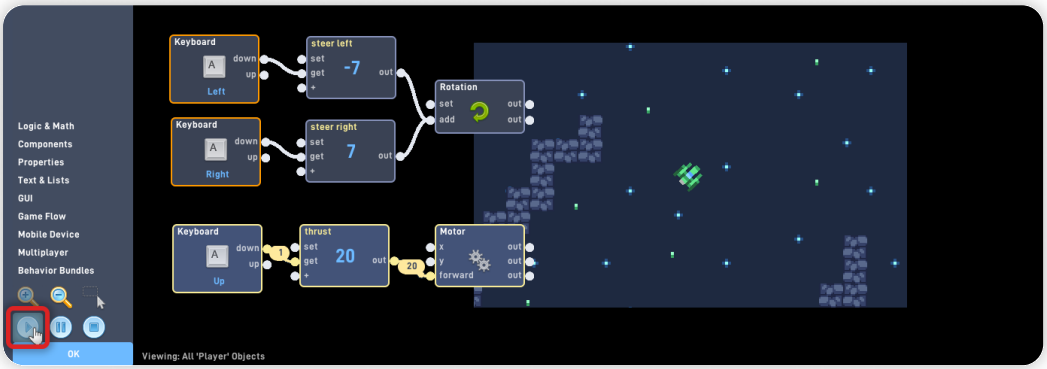
Click on the “thrust” Number behavior and change its “Current Value” from “6” to “20”. Click “OK” on the Number behavior panel to save your changes.


Now, let’s playtest and see how the updated “thrust”/”forward” force affects the spaceship movement and how it feels while playing the game.

Click on the “Play” button to playtest the game inside the editor.




*Perfect! Now the player spaceship moves faster and doesn’t float away in space when we are not applying movement forces. It already feels a lot better to play and control the spaceship.*

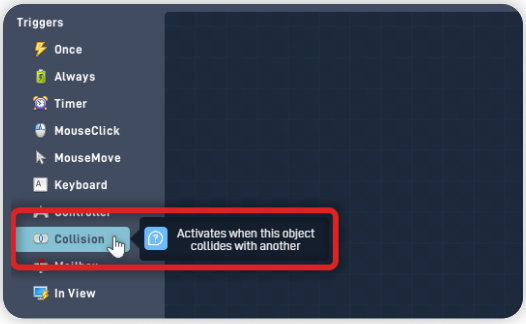
Click on the “Stop” button, near the “Play”, to stop the playtesting.
Click on the green “Check” button in the top left corner to close the bundle.

## Step 3

**Add Health Logic to the Player**

Still inside the “Player” behavior editor, click on the empty space to move the existing behaviors so you can see more empty area to add new Logic.

With the “Triggers” section open, add a “Collision” behavior.
Then, click and hold on to the newly added “Collision” behavior and move it to near the “Ship Controls” bundle.



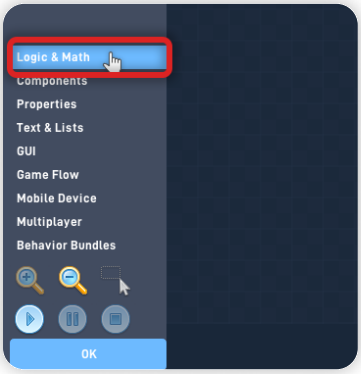


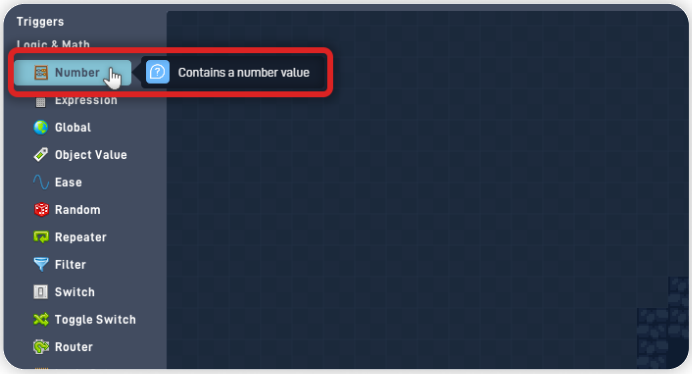
Click on the “Collision” behavior to open its behavior panel, and change the collision from “Any Type” to “Wall Rock”. *This change will make this behavior trigger everytime the “Player” object collides with the “Wall Rock” object.*

Change the collision “Repeat Delay” to “10” so it only detects new collisions ten frames after the previous collision.



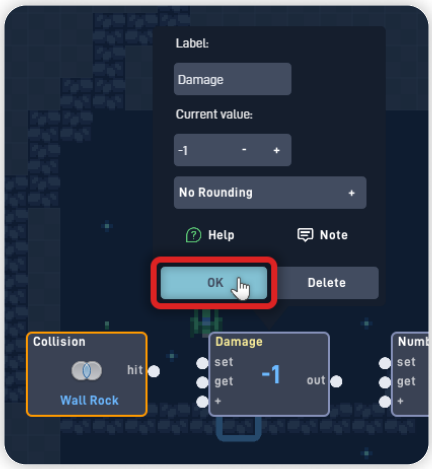

Click “OK” on the Collision behavior panel to save your changes.
Click on the “Logic & Math” section, and add two “Number” behaviors.



Again to move the newly added behaviors, click and hold to move and align the behaviors side by side.
Click on the Number, near the collision behavior, to open its behavior panel.


Set its “Label” to “Damage” and set its Current Value to “-1”.
Click “OK” to close the behavior panel and save your changes.

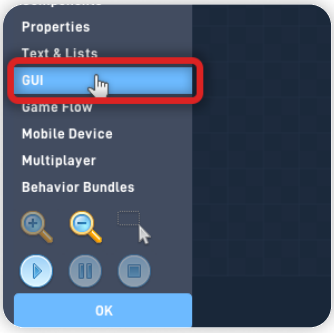

> The Logic triggers from left to right, so always organize your behaviors from left to right to **keep the Logic tidy and easy to read its flow.**
>
> If you add more behaviors than you need, you can remove by opening their respective behavior panel and clicking on the “Delete” button.

Click on the second Number to open its behavior panel.
Set its “Label” to “Health” and its “Current Value” to “3”.
Click “OK” to close the behavior panel and save your changes.

Now, click and hold on the “hit” output from the Collision behavior and connect it to the “get” input from the Damage Number.
Connect the Damage Number “out” to the “+” input of the Health Number.

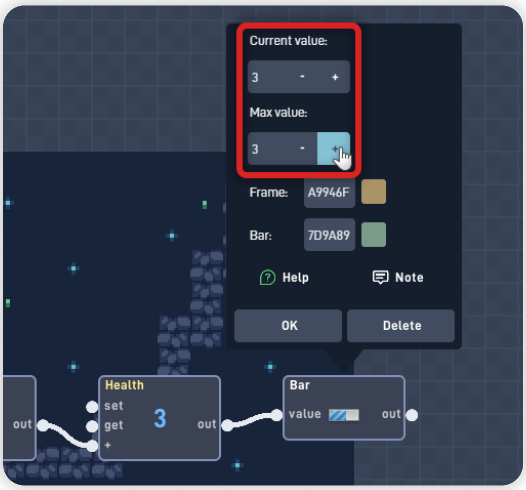

*This bit of Logic activates every time the Player collides with the “Wall Rock” object, reducing the “Health” Number value by 1.*

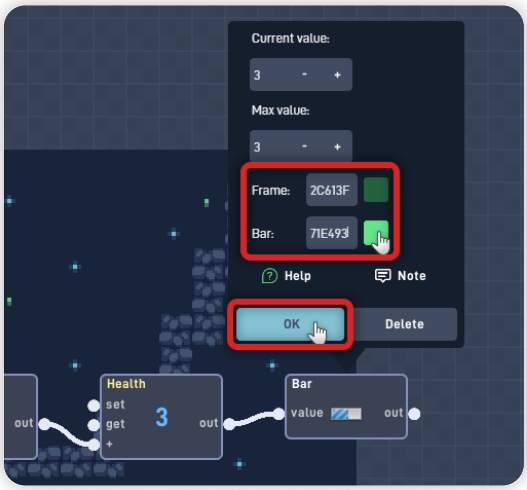
Click on the “GUI” section and add a “Bar” behavior.
Move the “Bar” behavior and place it near the Health number.
Connect the Health Number “out” to the Bar “value” input.






Open the “Bar” behavior panel by clicking on it, and change its “Current value” and “Max value” both to “3”.
Set the color for the “Frame” and “Bar” by clicking on the colored squares or typing the color HEX code in the fields.

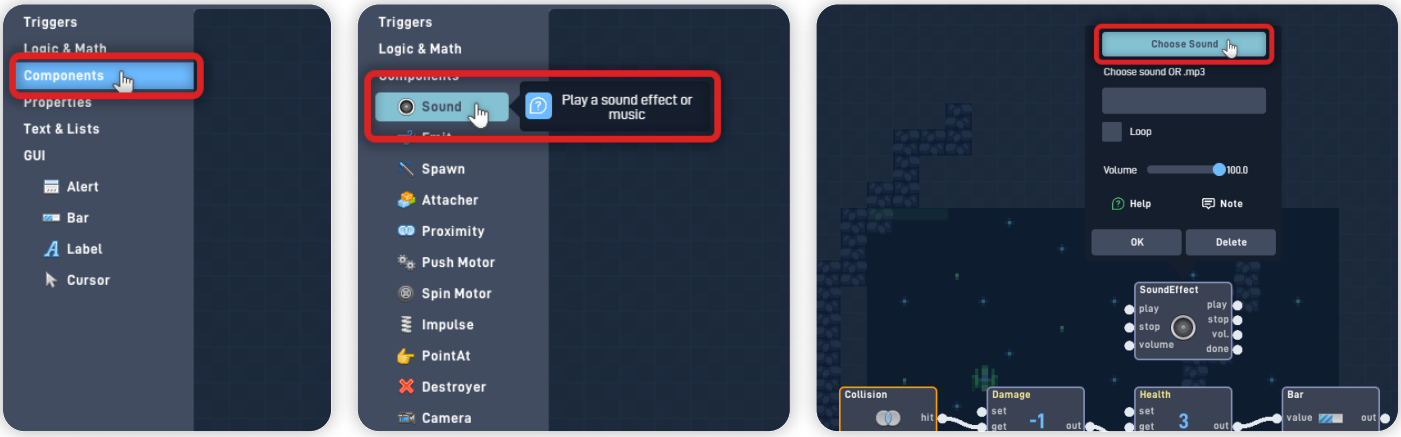



We’re using the player ship sprite colors for the bar, but you can set the colors that feel right to you. **Frame:** 2C613F (Dark Green) **Bar:** 71E493 (Light Green)

Click “OK” to close the behavior panel and save your changes.

Click on the “Components” section, add a “Sound” behavior, and place the newly added sound behavior above the “Health” number.
Click on the Sound to open its behavior panel, then click “Choose Sound”.




Inside this menu is where you can pick the Sound that will play when this behavior is triggered. Navigate the Library by clicking on the folders.

Go to “Effects”, then “Impact”, and select “**impact sound effect 1**”.
Click “OK” to confirm your sound effect.


Change this sound “Volume” to “30” by dragging the volume slider. *Always ensure your game’s sound effects aren’t too loud to give players a comfortable experience.*
Click “OK” to close the behavior panel and save your changes.

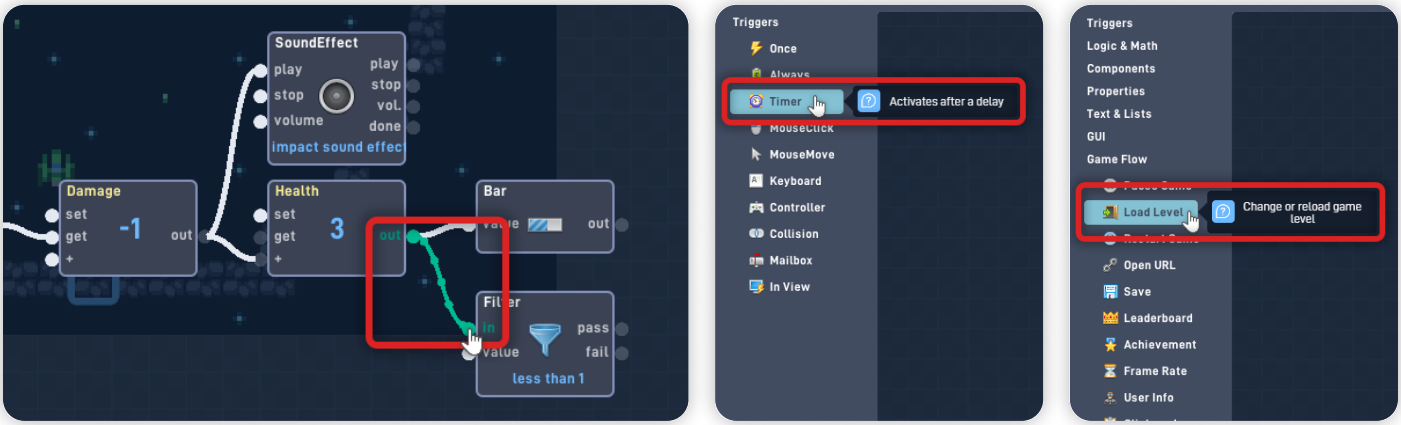
Now, connect the Damage Number “out” to the “play” input from the “Sound” behavior. *This connection makes the Sound play whenever the “Damage” number is activated, in this case, when the Player collides with the wall object.*


Click on the “Logic & Math” section, add a “Filter” behavior, and move it to below the Bar behavior.

Open the “Filter” behavior panel, and change its evaluate expression from “greater than” to “less than”.
Set the Filter value to “1”, and click “OK” close the behavior panel.

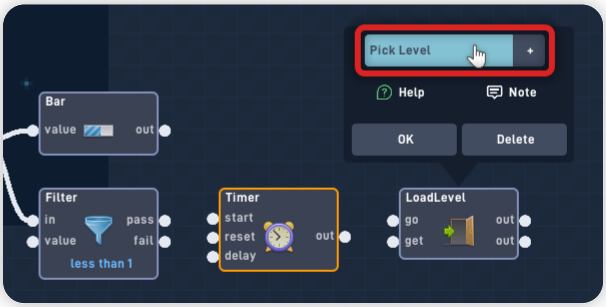

Connect the “Health” Number “out” to the “in” Filter behavior input.

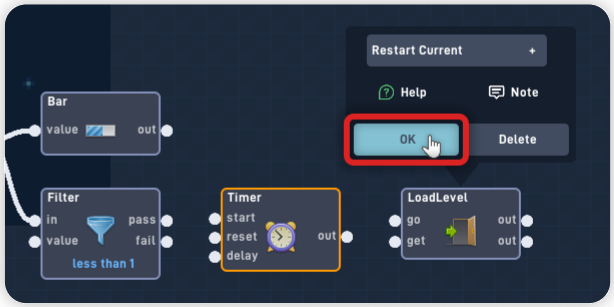
Open the “Triggers” section, and add a “Timer” behavior.
From the “Game Flow” section, add a “LoadLevel” behavior.



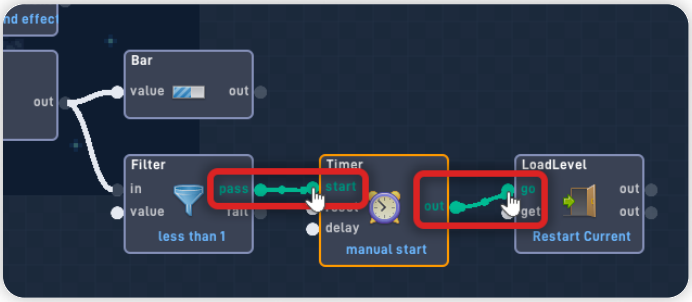

Move the newly added behaviors and place them side by side after the Filter.

Open the “LoadLevel” behavior panel, click on the “Pick Level”, and from the dropdown, set it to “Restart Current”.
Click “OK” to close the behavior panel and save your changes.

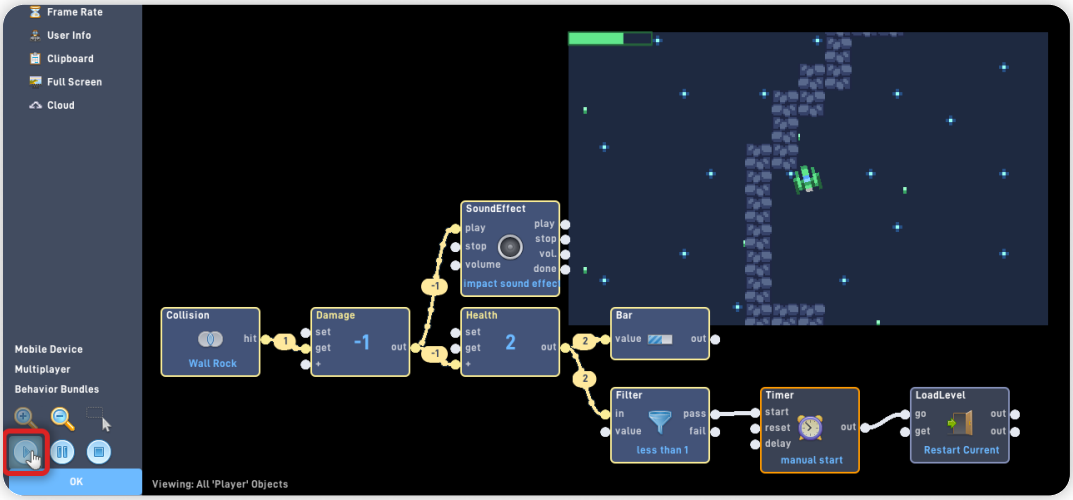



Connect the Filter “pass” output to the “start” input from the Timer behavior.
Connect the Timer “out” to the “go” input from the LoadLevel behavior.
*This newly added logic bit evaluates if the Health number value is “less than 1”:*
*If true, it will activate the timer and, one second later, will restart the current level.*




Move behaviors out of view so you can see the game below.
Click “Play” to playtest inside the editor.




*Fantastic! You added a lose condition to the game. It adds a consequence to moving around recklessly and makes the game more challenging.*

*While playtesting inside the editor, you can see that this new Logic triggers:*

- *Every time the Player collides with the wall, it plays a sound and reduces the Health value, displayed on the screen by the Bar User Interface element.*

- *If the Health value is less than “1”, it will restart the game level one second later.*

To stop playtesting, click on the “Stop” button near the “Play” button.

Click “OK” to close the Behavior Editor and save your changes.
Click “OK” to close the Player object properties panel.
Click “OK” again in the Library panel to close it.

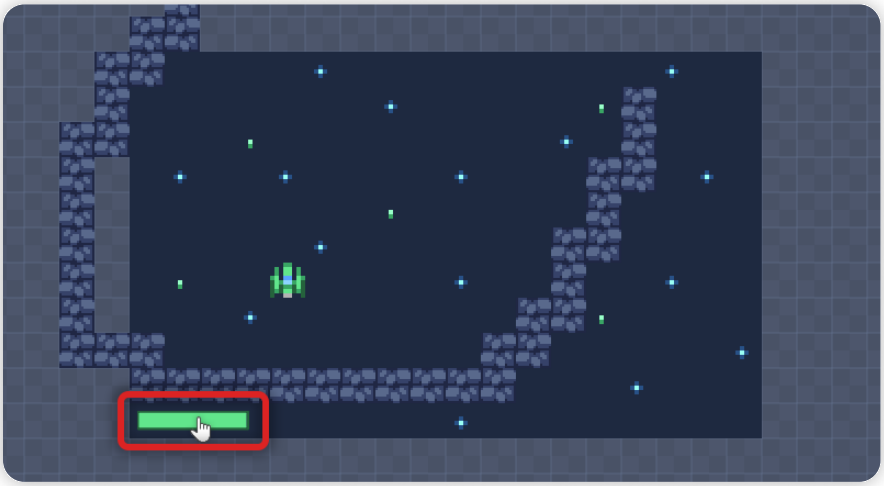
## Step 4

**Move the Bar on the User Interface layer**

Back in the Game Editor, click on the “Layer” button on the bottom toolbar to open the Layers panel.
Then, click on “User Interface” to change to the User Interface layer.


> Objects in the User Interface (UI) layer don’t have physics. This layer is **perfect for placing buttons, menus, or bars** because objects in this layer render on top of the game and don’t scroll with the camera.

Click and hold on to the Bar to move it around, and place it in one of the game’s view corners. *Remember that User Interface objects not inside the game’s visible area won’t appear when Playing.*



Now, click on the “Play” button in the bottom toolbar to play your game.

**When in Play mode:**

- The ship will feel better to control as it doesn’t drift away in space as in the previous lesson;
- When colliding with a wall, the Player ship plays a sound and loses Health which is displayed by the UI Bar;
- The game will restart if the Player’s health bar is empty - or, logically speaking, if the Health value it’s less than “1”;

*If you have problems, check the troubleshooting section.*

## Troubleshooting

A big part of game development is figuring out why things sometimes do not behave as you expect. If your game is misbehaving, check the following points:

- **If the Health bar doesn’t start full,** ensure that the Bar “Max value” is the same as the “Current value” and the Health number value; *(Step 3)*
- **If your game doesn’t restart at the right time** (when the health bar is empty), ensure that the Filter behavior is set to “less than 1”; *(Step 3)*
- **If colliding with the wall object doesn’t decrease the Health value,** make sure the “Damage” number “out” is connected to the “+” input from the “Health” number; *(Step 3)*

**Space Pilot - Part 1**

# Nice work!

You’ve now finished **Lesson 3 out of 6.**