# **Lesson 6:** Player Health UI & Damage

**Objective:** Create Health Points for the Player object and decrease this value when colliding with Enemies.
Keep track of the Player's Health using User Interface objects.

**Time:** 30 Minutes

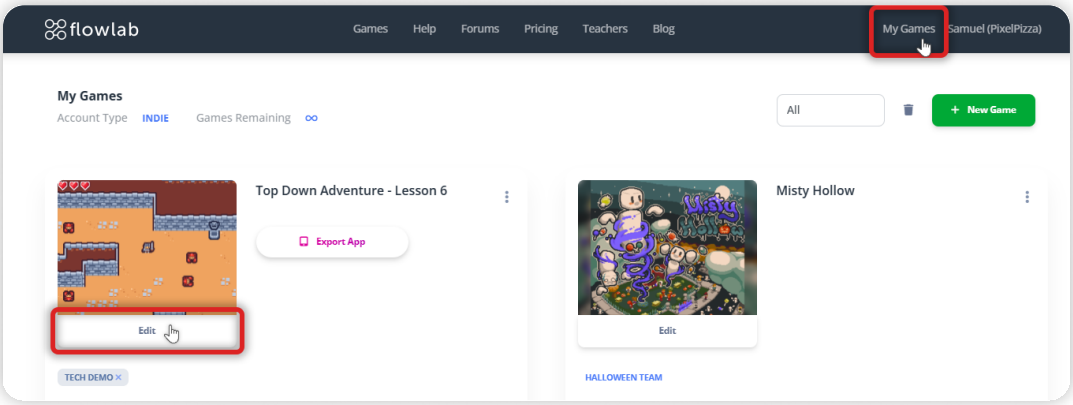
**Level:** 2.5 - Beginner/ Intermediate

**Description:** Using custom Logic, make the Player react and take damage when colliding with the Enemies. Using the User Interface layer, create Hearts representing the Player's Health. When the Player's Health reaches 0, the game restarts the current Level.

## Step 1

**Edit your Game**

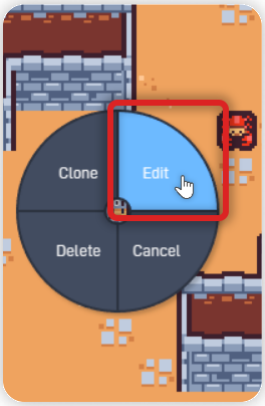
Log in and start at your "My Games" page https://flowlab.io/game/list
Then, click "Edit" below your game to open the game Editor.

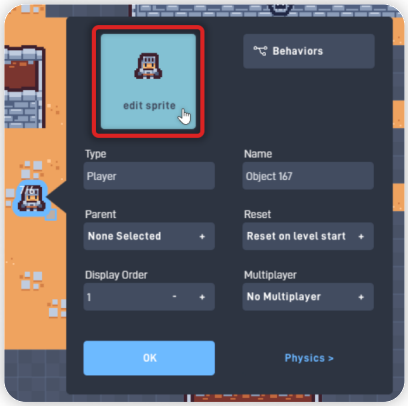


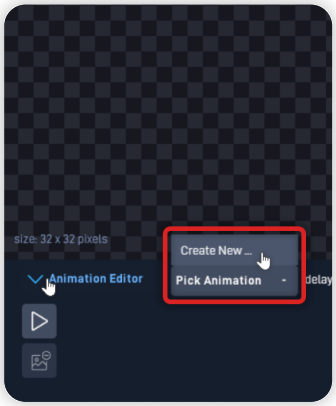

## Step 2

**Create a Damage Animation for the Player**

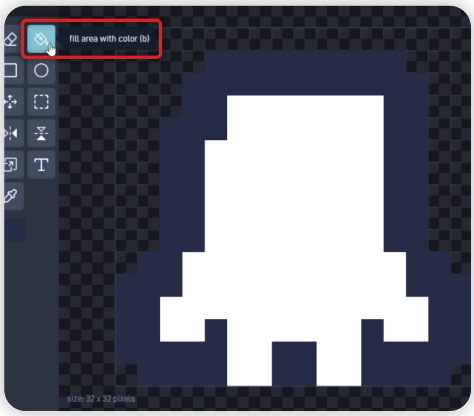
Click on the Player Object, select "Edit" from the circle menu, and click "Edit Sprite".
Click on the "Animation Editor" to open the Animation panel, click on "Pick Animation" and select "Create New...".
Set the Animation Name to "**Damaged**" and click "OK" to confirm.




Now, using the “Paint Bucket” tool, fill the Player Sprite with a White color and a brighter outline to create a Hit-Flash animation.



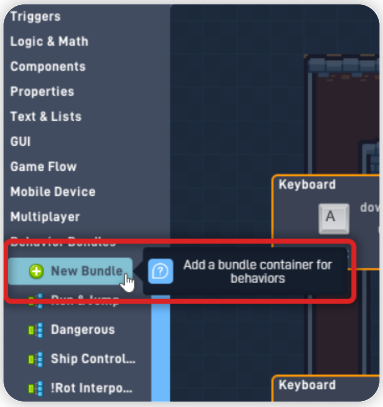

**Player Fill color:** FFFFFF (White)
**Brighter Outline Color:** 262B44 (Blue-ish Grey)

*Perfect, you've created a Hit-Flash animation to convey to the Player that their Character has taken damage.*
Click "OK" to close the Sprite Editor and save your changes.

## Step 3

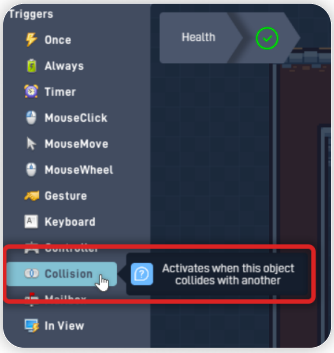
**Add Custom Health Logic to the Player**
With the Player Object properties panel open, click "Behaviors" to open the Behavior Editor.

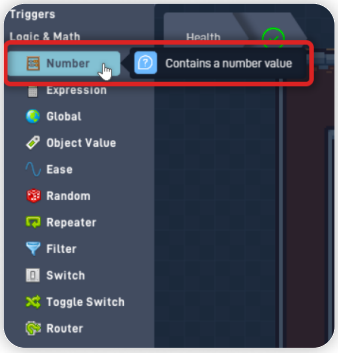
From the Behavior Bundles section, click **"New Bundle"** to add a new bundle.
Click and hold to move and place it below the other Bundles.
Click on the Bundle, set its Label to **"Health"**, and click "Open Bundle" to edit the Bundle content.

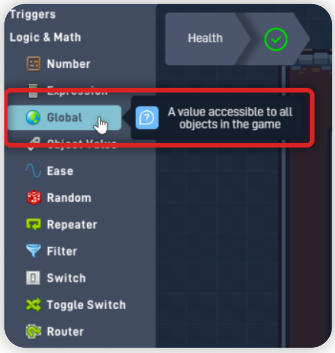



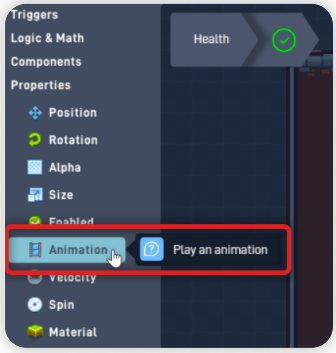


From the Triggers section, **add a "Collision"** behavior.
From the Logic & Math section, **add two "Number"** behaviors.
From the Logic & Math section, **add a "Global"** behavior.
From the Properties section, **add an "Animation"** behavior.
Organize the newly added behaviors as shown below.



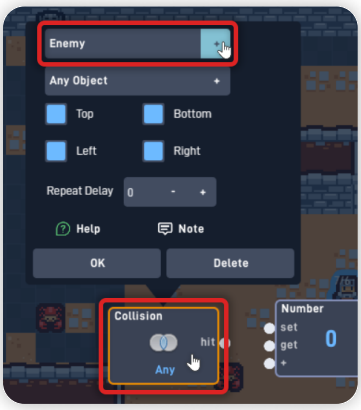



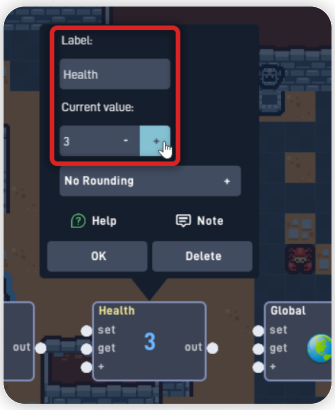

Open the Collision behavior panel and change its **target type to "Enemy"**.
Click "OK" to close the behavior panel and save your changes.

Open the first Number behavior panel, set its Label to **"Damage"**, and set its **Current value to "-1"**.
Click "OK" to close the behavior panel.

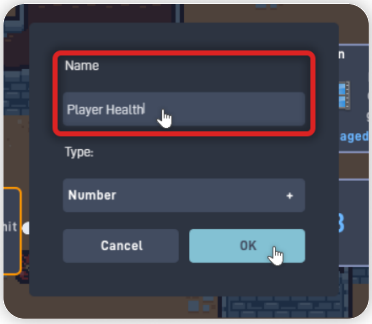
Open the second Number behavior panel, set its Label to **"Health"**, and set its **Current value to "3"**.
*This value sets how many times the Player can take damage until they lose.*
Click "OK" to close the behavior panel.








Open the Global behavior panel, click "Global Name", and select "Add New Name".
Set the Global Name to **"Player Health"** and click "OK" to confirm.
Click "OK" to close the behavior panel.

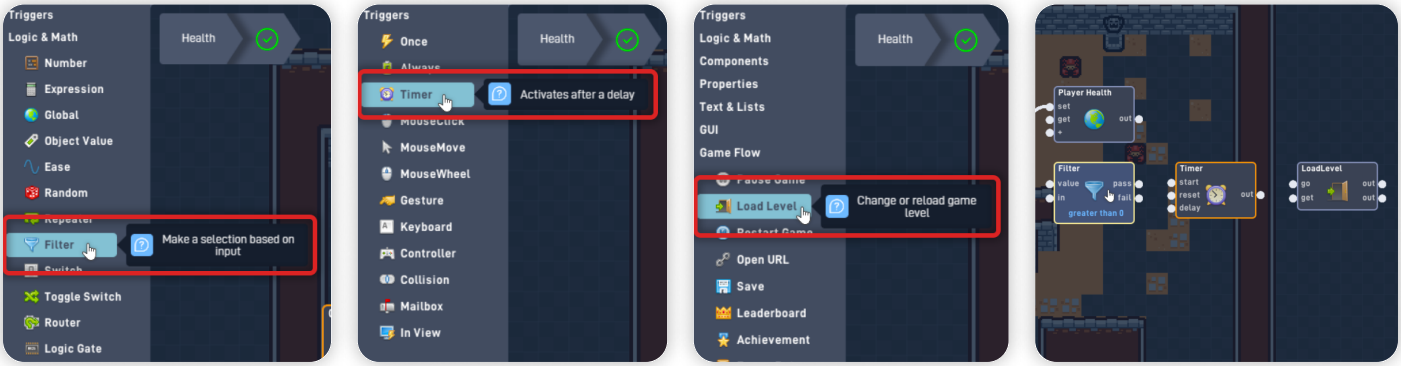




Now, let's connect these behaviors.
Connect the Collision "hit" to the "get" input from the Damage Number.
Connect the Damage Number "out" to the "start" input from the Animation behavior.
Connect the Damage Number "out" to the "+" input from the Health Number.
Connect the Health Number "out" to the "set" input from the Player Health Global.

*This new logic triggers every time the Player collides with the Enemy object.*
*The Collision activates the "Damage" Number, which plays a Hit-Flash Animation*
*and reduces the "Health" Value by 1.*

*The "Health" Number sets the "Player Health" Global value.*
*The "Global" Behaviors function as a global "broadcast" Message for all other objects.*
*Any object with that Global, will trigger and output their value every time its value is updated.*

From the Logic & Math section, **add a "Filter"** behavior.
From the Triggers section, **add a "Timer"** behavior.
From the Game Flow section, **add a "Load Level"** behavior.
Organize the newly added behaviors below the Global, as shown below.




Open the Filter behavior panel, set its formula to "less than" and set its value to "1".
Click "OK" to close the behavior panel.

Open the LoadLevel behavior panel, click "Pick Level", and select "Restart Current".
Click "OK" to close the behavior panel.

Now, let's connect these behaviors.
Connect the Health Number "out" to the "in" input from the Filter behavior.
Connect the Filter "pass" to the "start" input from the Timer behavior.
Connect the Timer "out" to the "go" input from the LoadLevel behavior.


*This last bit of logic evaluates the "Health" value each time it changes.*
*If the value is less than 1, it triggers a Timer to Restart the current Level because*
*the Player has lost all its Health Points.*

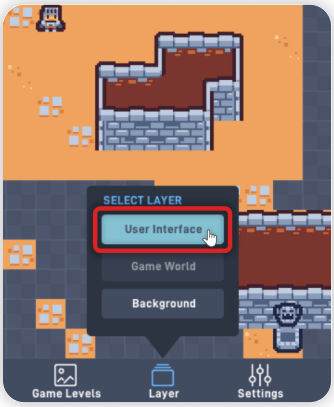
Click "OK" to close the Behavior Editor and save your changes.
Click "OK" again to close the object properties panel.

## Step 4

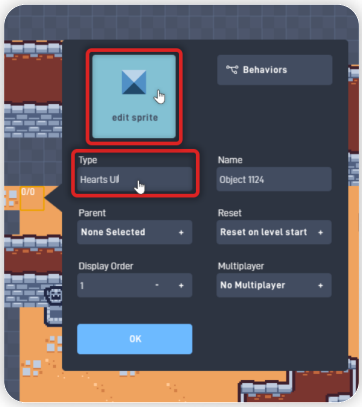
**Create the Health User Interface**

Click on the "Layer" button on the bottom toolbar, and switch to the "User Interface" layer.

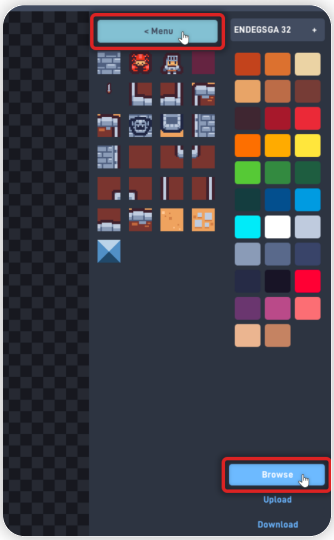
Click on the Top-Left corner from the game's viewable area, and select "Create" from the circle menu. Change its Type to **"Hearts UI"** and click "Edit Sprite" to open the Sprite Editor.

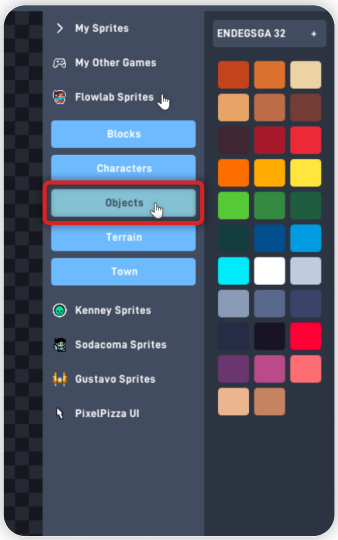


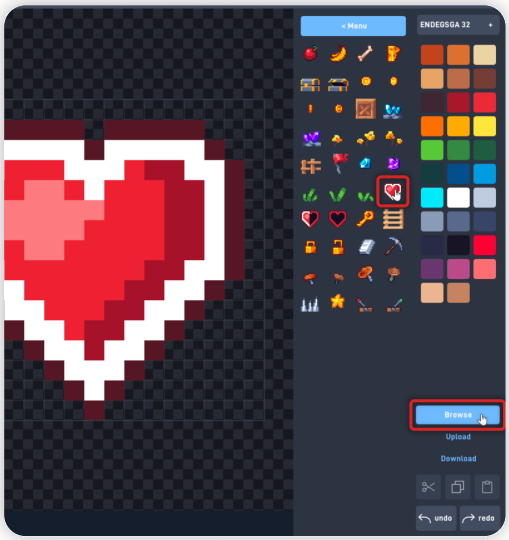




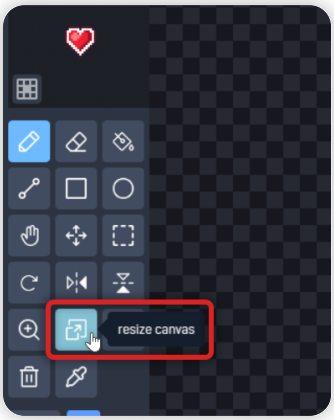

Click "Browse", then "< Menu", and from the "Flowlab Sprites" collection, select the "Objects" category. From this category, select the **"Filled Heart" Sprite** and click on "Browse" to close the browse panel.

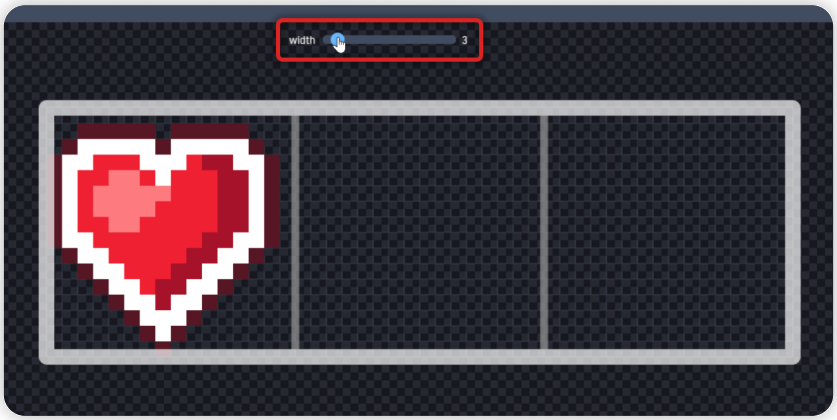


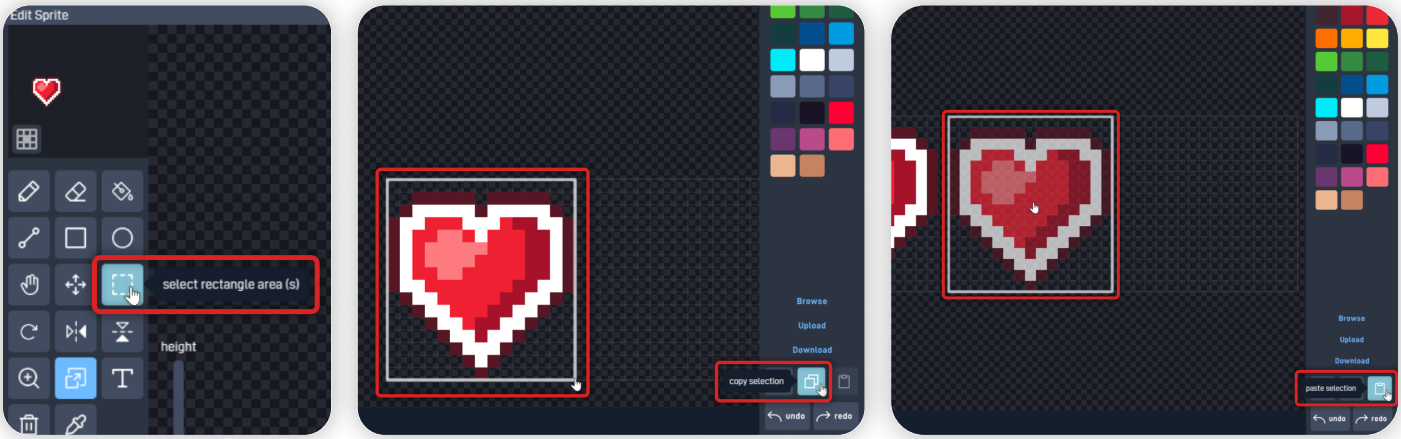

Select the "Resize Canvas" tool on the sidebar and set the Width to "3".



Change to the "Select Rectangle Area" tool. Click and hold from the Sprite's top-left corner and select the Heart Sprite as shown below.
Click on the "Copy" button on the right sidebar.

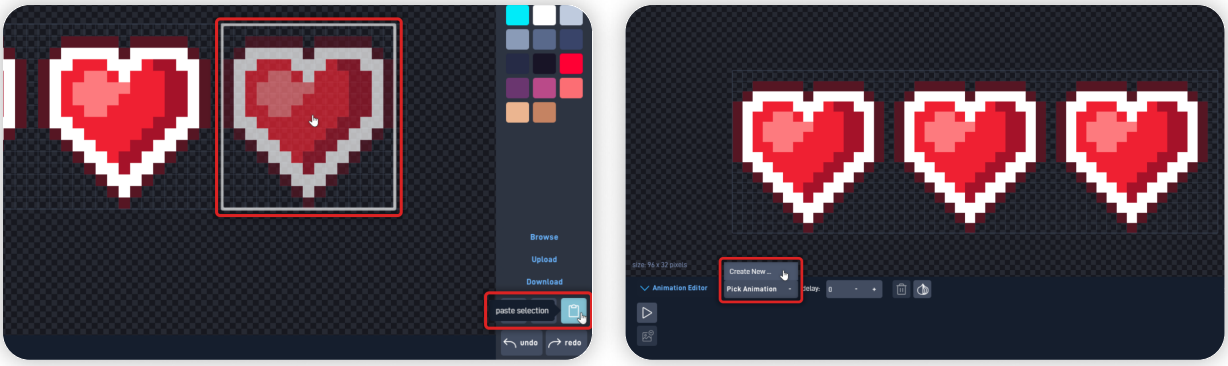
Drag the copied art to the right, and place it in the same position as shown below.
Once the preview is placed correctly beside the other Heart, click the "Paste" button.



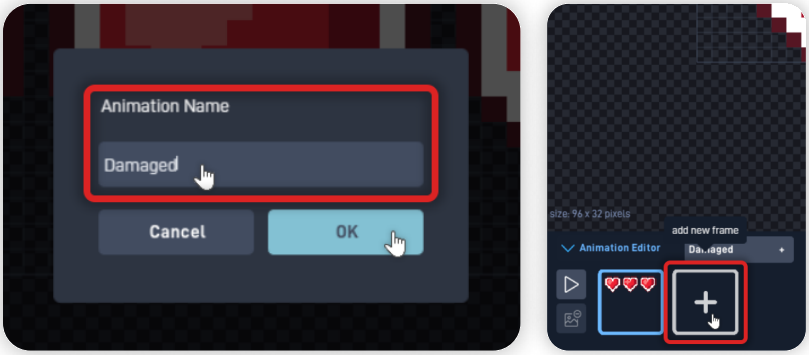

Drag the copied art to the right, and place it in the same position as shown below.
Once the preview is placed correctly beside the other Heart, click the "Paste" button.

Then, move the selection again, and move it to beside the second Heart.
Once in the correct position, click the "paste" button again.
*In the end, the Sprite should look as shown below, with three Hearts next to each other.*

Click on "Animation Editor" to open the Animation panel, then click on "Pick Animation" and select "Create New...".




Set the Animation Name to **"Damaged"** and click "OK" to confirm.
Click twice on the "+" Button to Duplicate the current frame and have three repeated frames in total.

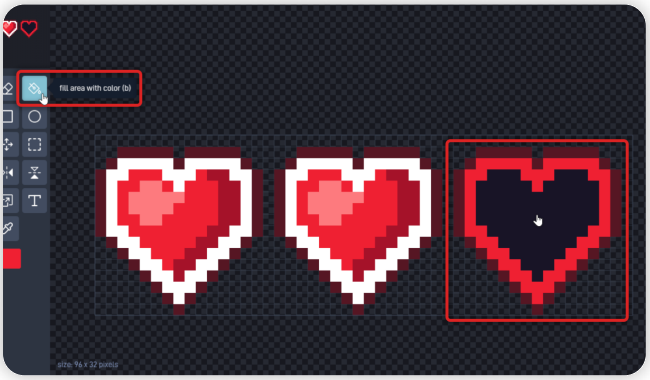


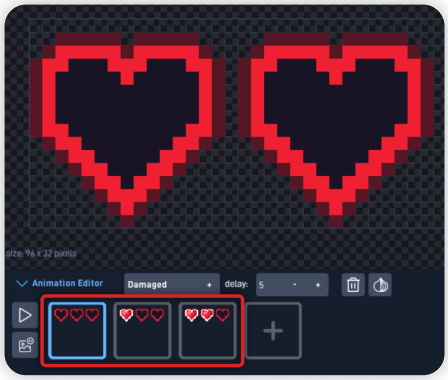
Select the "Paint Bucket" tool on the sidebar, and select a Dark Color to fill the inside of the Heart.
With the third frame selected, fill the inside of the right-most Heart.
Then, using the Heart's Red color, change the white outline to Red, as shown below.

**Empty Heart Color:** 181425 (Dark Purple-ish Black)
**Empty Heart Inner-Outline:** E83641 (Red)

Repeat this step and fill the other Hearts in the previous frames, as exemplified below.
*In the end, the first frame must have all Hearts painted empty,*
*the second frame painted with two empty Hearts,*
*and the third frame painted with one empty Heart.*

Click "OK" to close the Sprite Editor and save your changes.

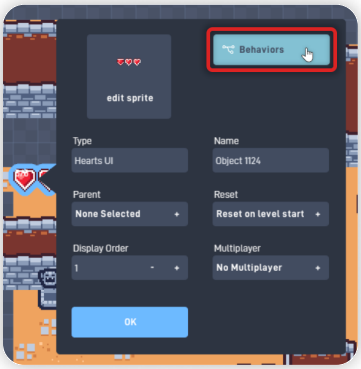


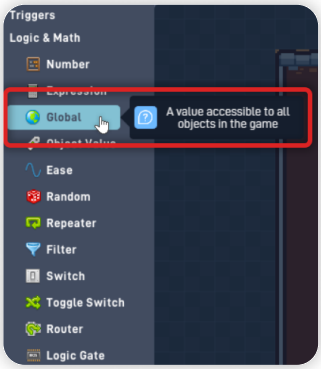



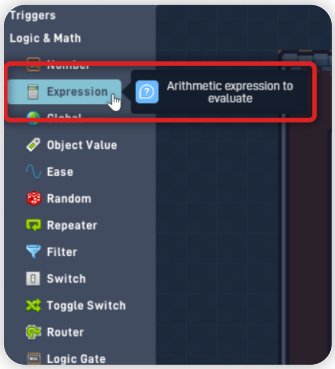

## Step 5

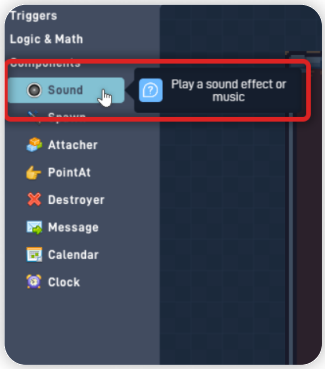
**Add Custom Logic to the Hearts UI**
Still with the Hearts UI properties panel open, click "Behaviors" to open the Behavior Editor.

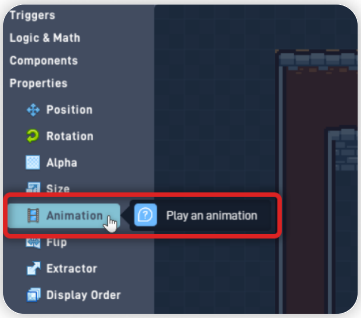
From the Logic & Math section, **add a "Global"** behavior.
From the Logic & Math section, **add an "Expression"** behavior.
From the Components section, **add a "Sound Effect"** behavior.
From the Properties section, **add an "Animation"** behavior.



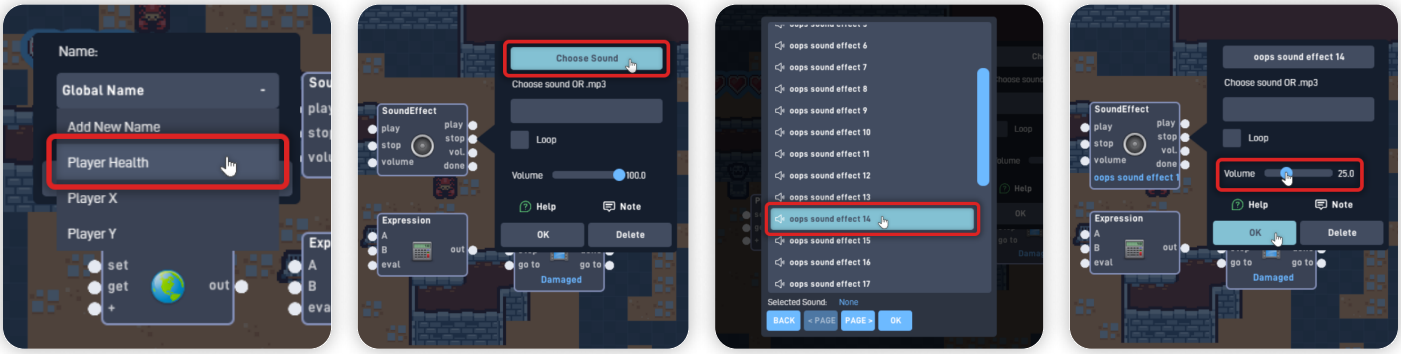



Organize the newly added behaviors as shown below.


Open the Global behavior panel and select its Name to **"Player Health"**.

Open the SoundEffect behavior panel and click "Choose Sound".
Navigate through the following folders: Effects > Oops.

From this folder, select **"Oops sound effects 14"** and click "OK" to confirm your selection.
Change the **Volume to "25**" and click "OK" to close the behavior panel.




Open the Expression behavior panel, set its Label to "Select Frame", and in the f() expression box, **Type in "A+1".**
Click "OK" to close the behavior panel.

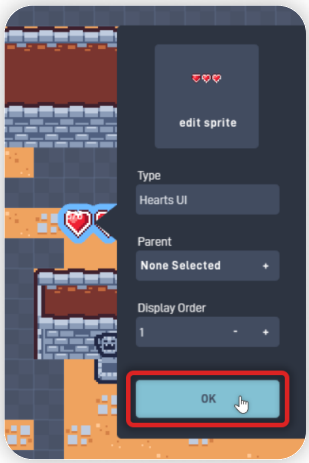
Now, let's connect these behaviors.
Connect the Global "out" to the "play" input from the SoundEffect.
Connect the Global "out" to the "A" and "eval" inputs from the Expression.
Connect the Expression "out" to the "go to" input from the Animation.

*This logic triggers every time the Player Health value changes or is updated.*

*Once the Global triggers, it plays an "Oops" sound effect to communicate to the Player that they have taken damage.*

*The Global value is also input through the Expression, which outputs the Frame number (1st, 2nd, or 3rd frame) to select a frame from the "Damaged" Animation.*

***For Example:** If the Player Health Global outputs 2, the Expression outputs "3" and plays the third frame of the "Damaged" Animation.*
*The 3rd frame of the this animation shows "Two filled Hearts and one empty Heart" to symbolize that the Player only has 2/3 left of their Health Points.*




Click "OK" to close the Behavior Editor and save your changes.
Click "OK" again to close the object properties panel.

## Step 6

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


**When in play mode:**
- When the Player object collides with the Enemy object, 1 value is subtracted from the Health Number;

- The Hearts UI object displays the Player Health using an Animation - The Player Health" Global value dictates the frames of the animation;

- The Hearts UI object stays in the same position while the Camera-View scrolls because it is an object from the User Interface layer;

*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 expected. If your game is misbehaving, check the following points:

- **If the Hearts Display doesn't correspond to the Player Health value,** ensure the frames are painted correctly and in the correct order; *(Step 4)*

- **If the Player Health isn't decreasing when taking damage,** ensure the Damage Number is connected to the "+" from the Health Number on the Player Logic; *(Step 3)*

- **If the Level isn’t restarting upon losing all the Health,** ensure that on the Player logic, the Filter formula is set to “less than 1”; *(Step 3)*

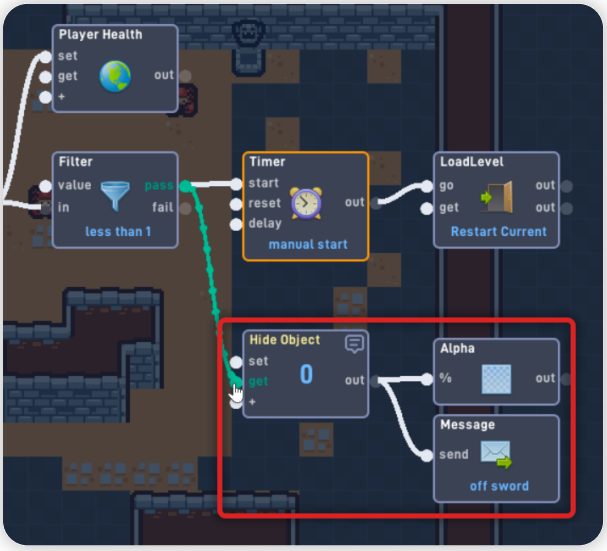
## Optional Game Enhancements

Once you have this Lesson's content working, here is a simple enhancement to try:

- **Hide the Player object when it has lost all its Health:**
  *By adding additional logic to the Player's "Health" bundle, you can make the Player object turn invisible upon losing, helping convey to the Player that they lost the game.*

  Create this additional logic by adding a "0" Number connected to an Alpha behavior to turn the Player object invisible.
  Also connected to the "Hide Object" Number, add a Message behavior, set its name "off sword", and select "Send to myself".

  Then, Inside the "Sword Logic" bundle, add a Mailbox behavior with "off sword" as its name and connect it to the "off" input from the Attacher behavior.





**Top Down Adventure - Part 1**

# Nice work!

You've now finished **Lesson 6 out of 6.**