Lesson 3: Enemy & its Movement Logic

Objective: Create an Enemy object, using Lists and custom Logic make it independently move in a Random Direction.

Time: 30 Minutes

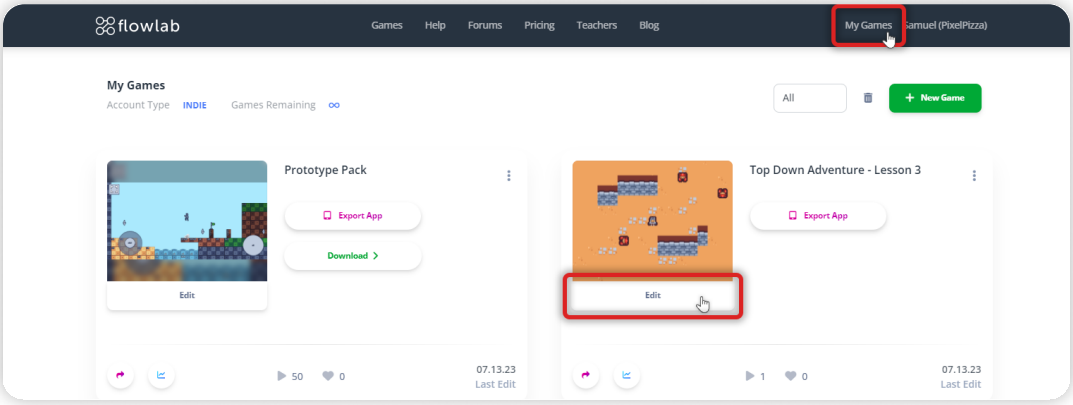
Level: 3 - Intermediate

Description: A basic Introduction to “Parents”, “Lists Data Types”, and List behaviors - An useful behavior for storing multiple values in a single container.

Step 1

Edit your Game

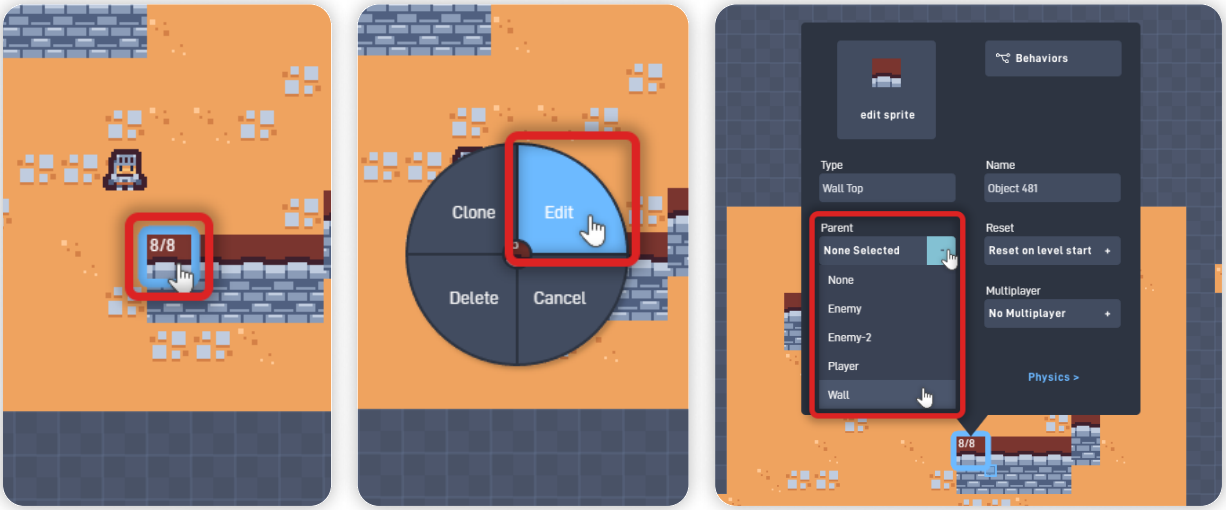
Log in and start at your “My Games” page https://flowlab.io/game/list
Then, click “Edit” next to your game to open the game editor.



Step 2

Set a Parent to “Wall Top” Object

Inside the Game’s Level editor, click on the “Wall Top” object and select “Edit” from the circle menu.
With the Wall Top Object Properties panel open, **set its “Parent” to “Wall”.**



Parent Objects:
Setting a Parent to an object makes it inherit the Logic and Collisions from the Parent object, a helpful feature that saves you time and makes editing multiple objects easier.

Since the Wall object has no logic, their Children objects won't inherit logic. But, Raycasts and Collisions set to "Wall" will also trigger when colliding with the "Wall Top" object.

When an object has a parent:
1. The Parent's behaviors run first, then the Child's.

2. Collisions (or Proximity) behaviors set to trigger with the Parent object will also trigger with its Children.

3. Child Objects will also receive messages sent to the Parent.

Note: An object can only have one Parent, but Parents can have any number of Children.

Parent Objects share their Collisions & Logic

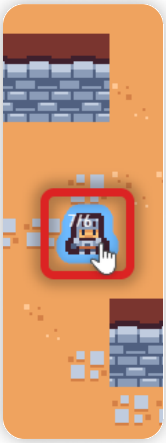
In the field of Computer Science, sharing logic this way is called "Inheritance".
Programming by creating and composing objects is called "Object-Oriented" programming.

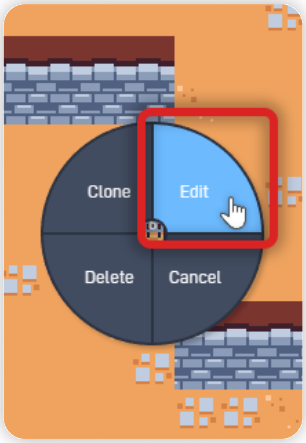
Click “OK” to close the “Wall Top” Object Properties panel and save your changes.
If you have more Wall objects, repeat this step to any other Wall objects you created, except for the “Wall” object itself.

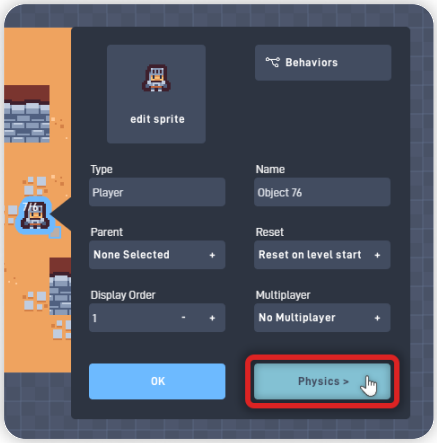
Step 3

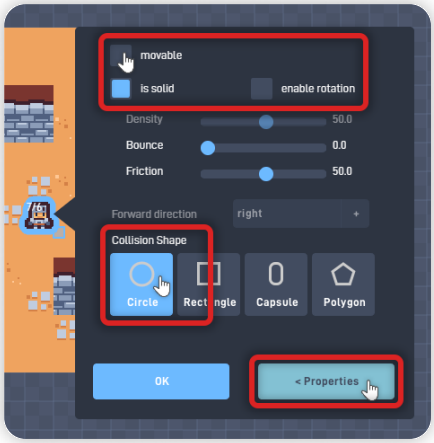
Update the Player Object Physics and Behaviors
Click on the “Player” object, and select “Edit” from the circle menu to open their Object Properties panel.

Click on “Physics >” to go to the Physics tab.
Then, unselect “Movable” and set the Collision Shape back to “Circle” again.







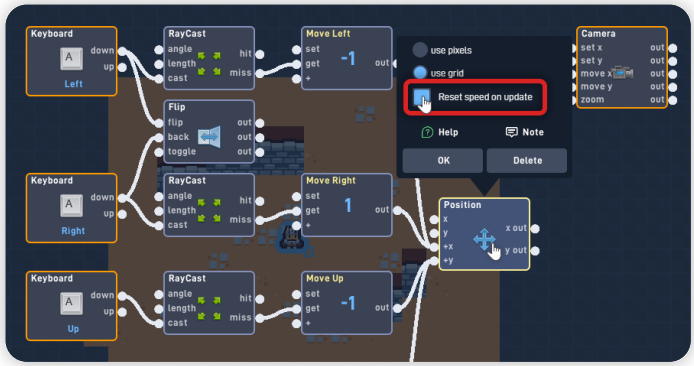


This change makes this object only movable through forces applied through its Logic and Behaviors (Position, Velocity, etc.), so this change makes the object stay in place if another object "bumps" into it.
Click on "< Properties" to return to the Properties tab.

Click on "Behaviors" to open the Player's Behavior Editor.

Click on the Position behavior to open its behavior panel, and select "Reset speed on update", so this object resets any force/speed after moving.

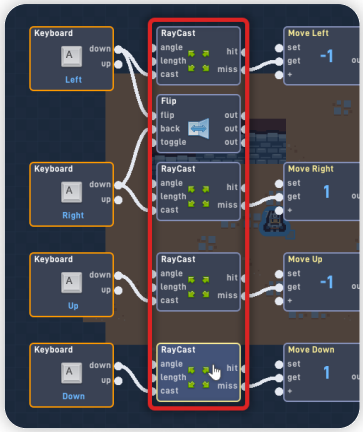
Click "OK" to close the Position behavior panel.

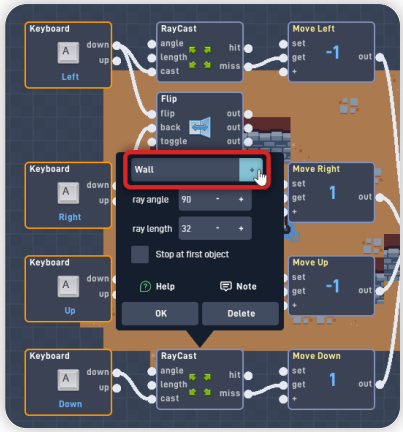


Now, click on the highest RayCast behavior to open its behavior panel, and change its Target Object Type from "Any" to "Wall".

*This change makes it so the **Player can only move if there isn't a Wall or its Children objects** (the "Wall Top" object and the "Enemy" object we will create on the next step) in front of it.*

Repeat this Raycast step with the remaining 3 Raycasts behaviors until all Raycasts are set to "Wall".





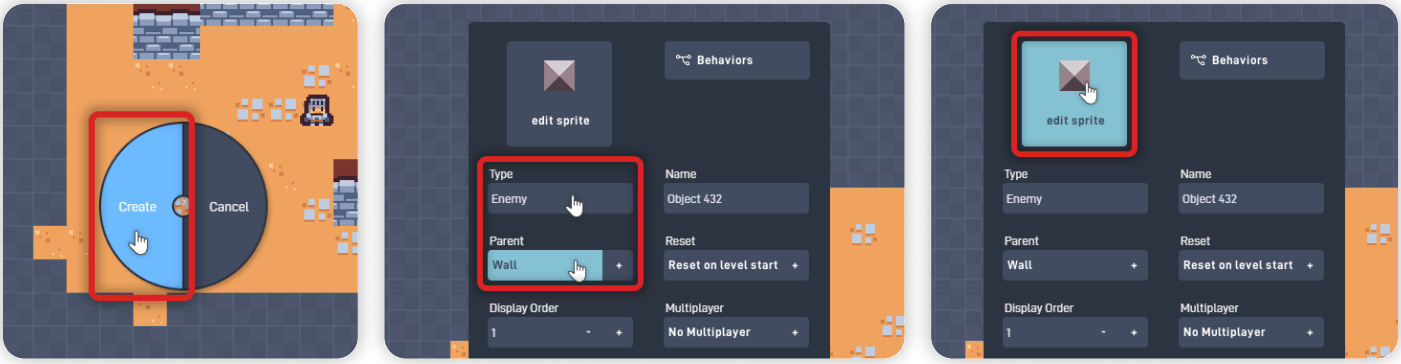
Click "OK" to close the Behavior Editor and save your changes.
Click "OK" again to close the Player Object Properties panel.

Step 4

Create an Enemy Object

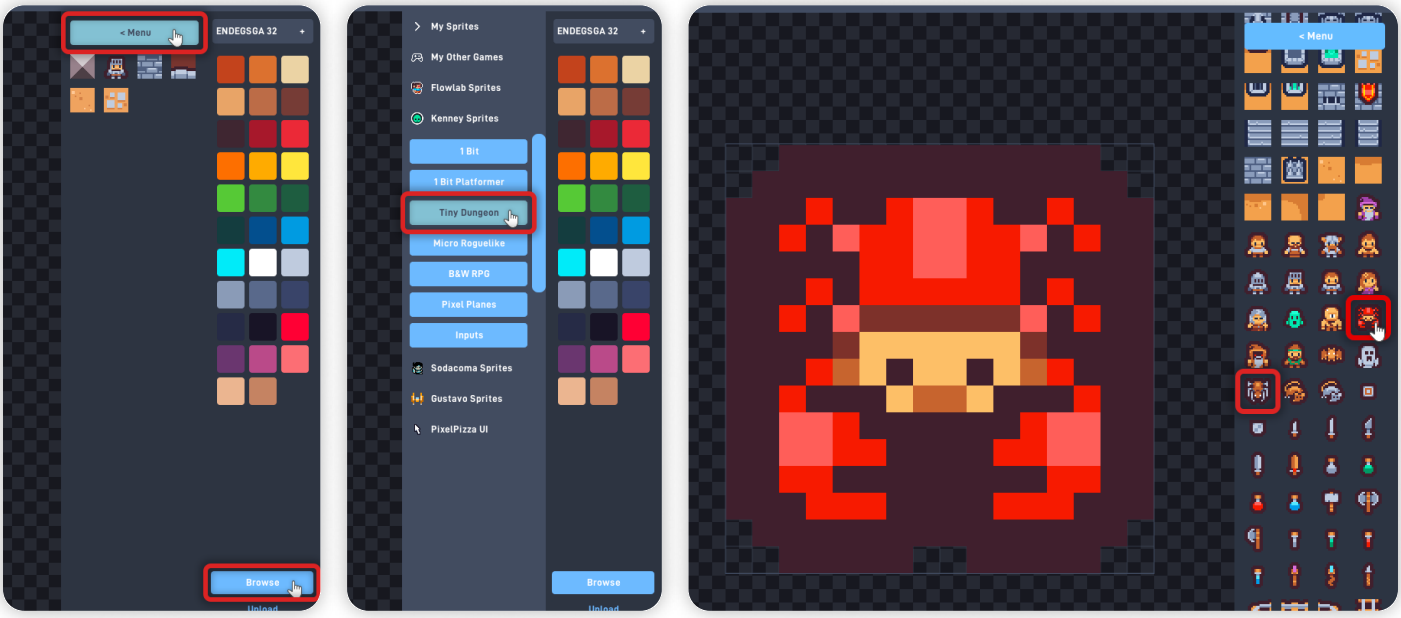
Click anywhere inside the visible game area, and select “Create” from the circle menu that appears.

It’s always important to name our objects, so let’s set this object “Type” to “Enemy”.
Set this object **“Parent” to “Wall”**.
This change makes the Enemy object inherit the “Wall” object collisions and makes the Player unable to move when in front of this object (due to its Movement Raycasts being set to Wall).



Now, click “edit sprite” to open the Sprite Editor and change this object sprite.

Inside the Sprite Editor, open the “Browse” panel and click on “< Menu” to open the Sprite Collections.
From the “Kenney Sprites” collection, open “Tiny Dungeon”.
Select the “Red Crab” or the “Spider” sprite.

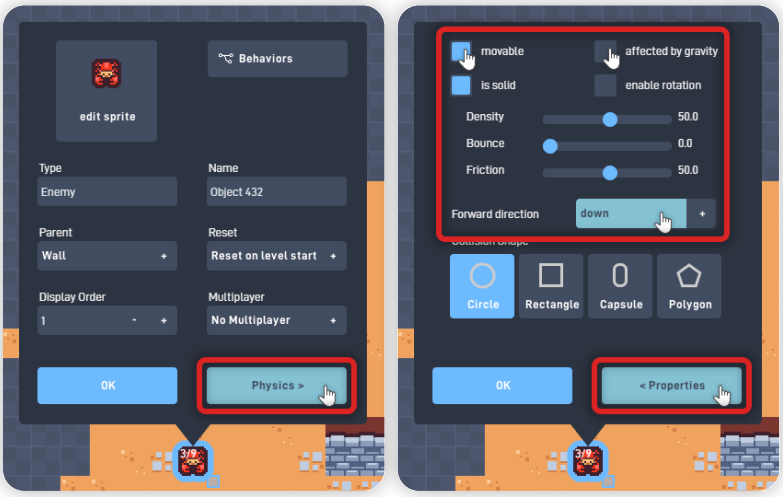


This Sprite will be used for the Enemy our Player must avoid and fight against, which is taught in our following lessons.

Once you pick your sprite, press “OK” to save your changes and close the Sprite Editor.

Now, click “Physics >” to go to the Physics tab.
Select “movable” and **unselect “affected by gravity”**.
Change its forward direction from “right” to **“down”**.

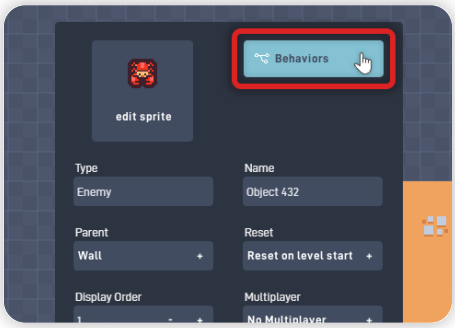
Click on “< Properties” to return to the Properties tab.



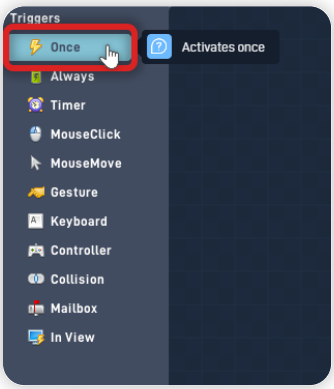
Step 5

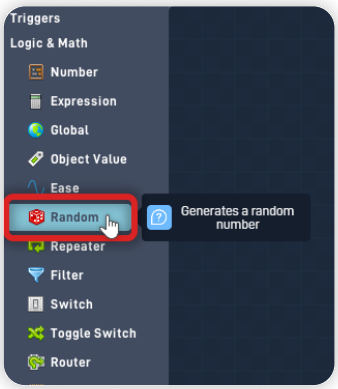
Make the Enemy Move using Custom Logic

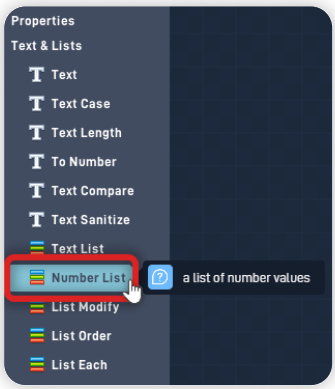
Click “Behaviors” to open the Enemy’s Behavior Editor.

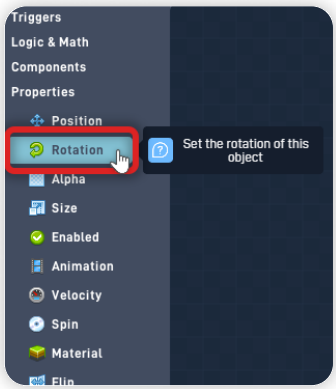


From the Triggers section, add a “Once” behavior.
From the Logic & Math section, add a “Random” behavior.
From the Text & Lists section, add a “Number List” behavior.
From the Properties section, add a “Rotation” behavior.









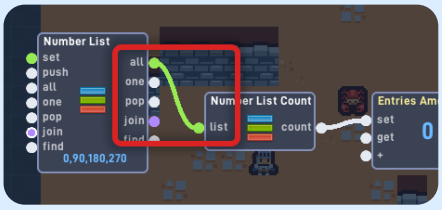
As shown below, organize the newly added behaviors from left to right by their added order.

Connect the Once “out” to the “new” input from the Random behavior.

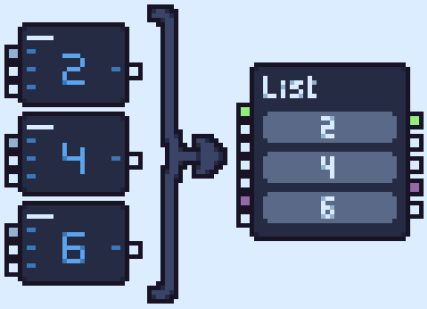
Click on the “Random” behavior to open its behavior panel and set its Min value to “1” and its Max Value to “4”.
Click “OK” to close the Random behavior panel.

Then, click on the “Number List” behavior to open its behavior panel.
Ensure the value is set to “0” and click “Add” to add this value to the List.
In the Number Area, you can type or set the value to be inserted into the List.

List Behaviors and List Data Types:
So far, we’ve been using behaviors that work with Number Data Types (White inputs/outputs), now let’s learn about Lists, a different Data Type. *In Flowlab, Lists are represented by the* **Bright Green Inputs/Outputs.**



Lists are able to store multiple values in a single container.
They are helpful to simplify and streamline logic and reduce the number of behaviors needed to store values.



In the field of computer science, a List is an example of something called a "Data Structure" - A way of storing individual values so you can easily access and use them later.

We often use lists in daily life, such as Grocery Lists or To-Do Lists.
In this case, we will store a list of rotation angles to make it easy to select one at random whenever we need it.

Help

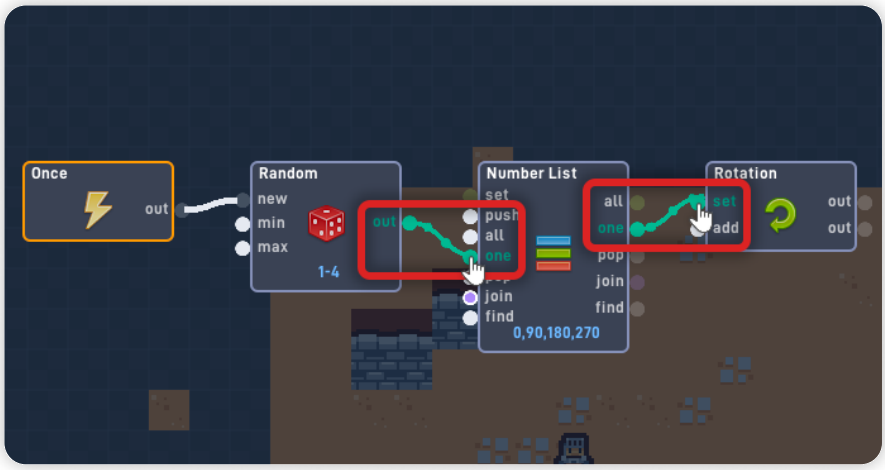
Learn more about Behaviors with the Help Button:
Remember, you can always learn more about each Behavior functionality and all its inputs and outputs through the Documentation accessible through the “Help” button inside each Behavior panel.

It includes a helpful Example video and a detailed description of its functionality and how to use it.

Now, type “**90**” in the value area and click “Add” again to add this new value to the List.
Repeat this step and add “**180**” and “**270**” as items to your List.
In the end, your List behavior panel should appear as shown below.
If you accidentally added an extra Item/Entry, you can delete it by selecting it on the List display and clicking “Removed Selected Item”.

Click “OK” to close the Number List behavior panel.

Connect the Random “out” to the “one” input from the Number List behavior.
Connect the Number List “one” output to the “set” from the Rotation behavior.

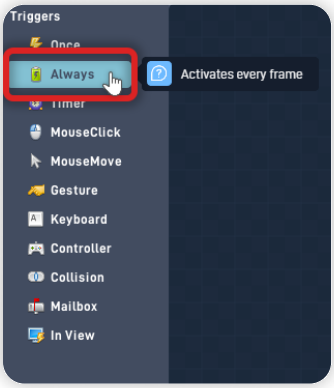


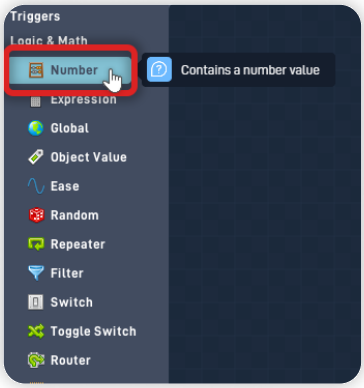
This bit of logic triggers Once the game starts, generating a Random number from 1 to 4.
The Random result is input into the Number List, selecting an entry based on its position (1st to 4th).
Then the List outputs the entry (Rotation Angles items we added) and sets this object Rotation.

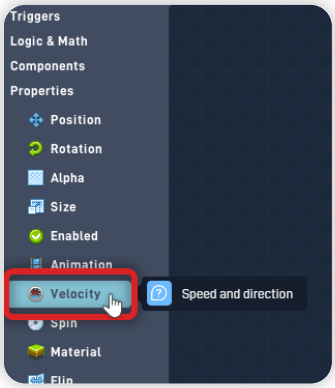
***Example:** Inputting “1” into the “one” input from the Number List will output the entry at the first position.*

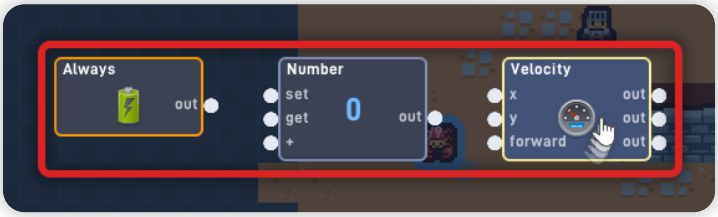
Let’s make this Enemy object move forward in their facing direction.
From the Triggers section, add an “Always” behavior.
From the Logic & Math section, add a “Number” behavior.
From the Properties section, add a “Velocity” Behavior.

As shown below, organize the newly added behaviors from left to right by their added order.



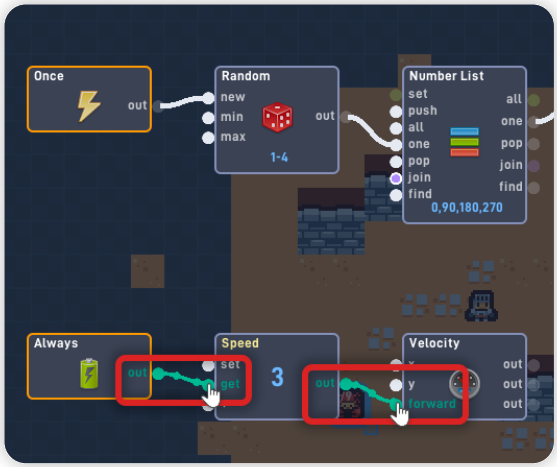






Click on the “Number” behavior to open its behavior panel, set its Label to “Speed”, and set its Current Value to “3”.
Click “OK” to close the Number behavior panel.

Now connect these behaviors together.
Connect the Always “out” to the “get” input from the Number behavior.
Connect the Number “out” to the “forward” input from the Velocity behavior.

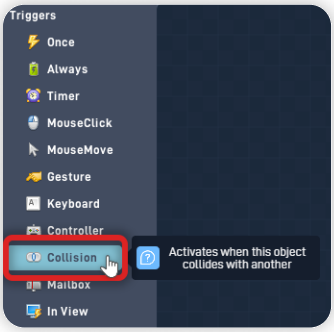


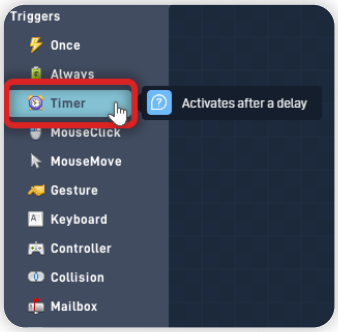
This new bit of logic triggers every frame setting this object's Forward Velocity to 3, making it move slowly in the direction the Sprite faces.

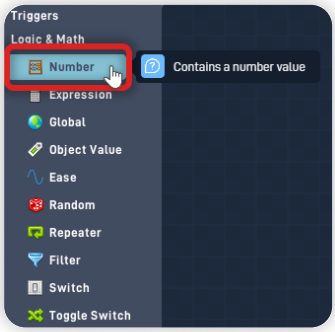
Now, let’s make this Enemy object turn around and move in the opposite direction after a while or after colliding with a Wall object.

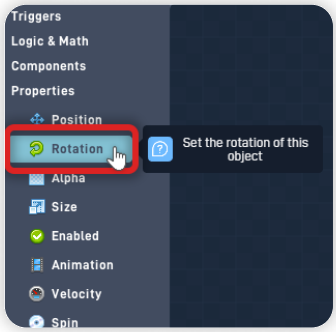
From the Triggers section, **add a “Collision”** behavior.
From the Triggers section, **add a “Timer”** behavior.
From the Logic & Math section, **add a “Number”** behavior.
From the Properties section, **add a “Rotation”** behavior.

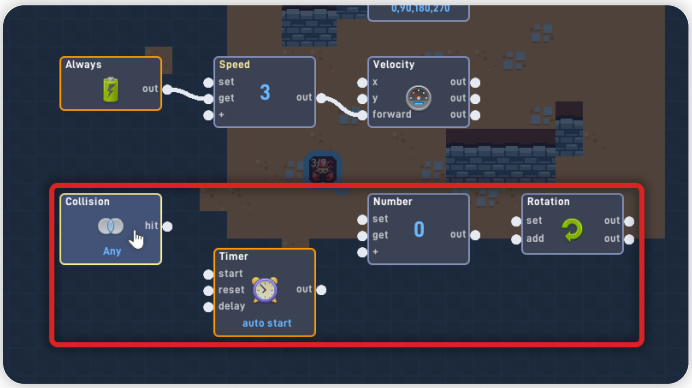
As shown below, organize the newly added behaviors by their added order.

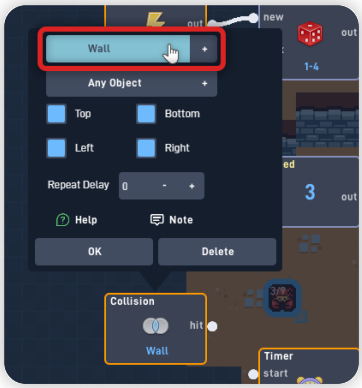












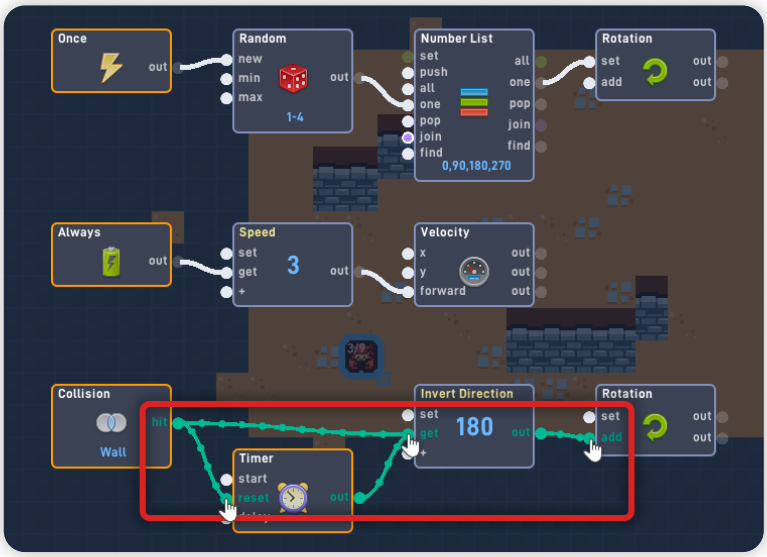
Click on the “Collision” behavior to open its behavior panel, and change the Object Type from Any Type to “Wall”.
Click “OK” to close its behavior panel and save your changes.

Click on the “Timer” behavior to open its behavior panel.
Set its Delay to “15”, select “Repeat Forever”, and click “OK” to close its behavior panel and save your changes.

Click on the “Number” behavior to open its behavior panel, set its Label to “Invert Direction”, and set its Current Value to “180”.
Click “OK” to close its behavior panel and save your changes.

Now let’s connect these behaviors together.

Connect the Collision “out” to the “reset” input from the Timer behavior.
Connect the Collision “out” to the “get” input from the Invert Direction Number behavior.
Connect the Timer “out” to the “get” input from the Invert Direction Number behavior.
Connect the Invert Direction Number “out” to the “add” input from the Rotation behavior.



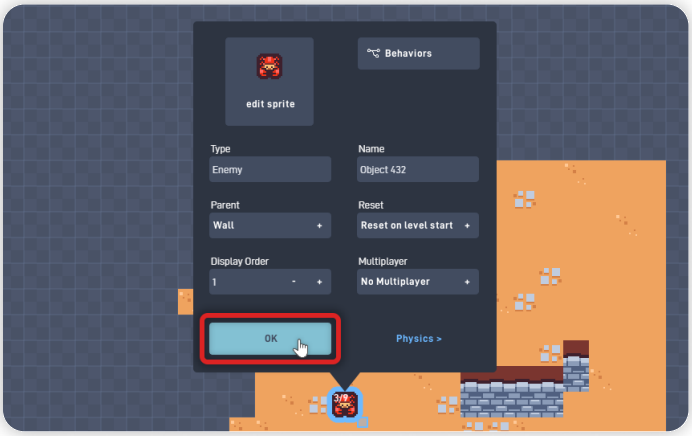
This new bit of logic triggers when this Object collides with a Wall object, making it turn 180° degrees and consequently make it move in the opposite direction.

This logic also triggers with the Timer, which “auto-starts” once the game starts, and after 15 frames, it activates the Number and Rotates this Object 180° degrees.

The Collision behavior also restarts the Timer to prevent this logic from triggering twice (which would make this object rotate twice, thus making it keep its original rotation and "get stuck" until the timer activated again).
This "double-trigger" is still bound to happen due to the simplicity of the code, but it won't happen as frequently.

Fantastic work! We've finished creating the logic for an Enemy object that picks a random direction once the game starts and keeps moving in a "bi-directional loop" (up and down or left and right).

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

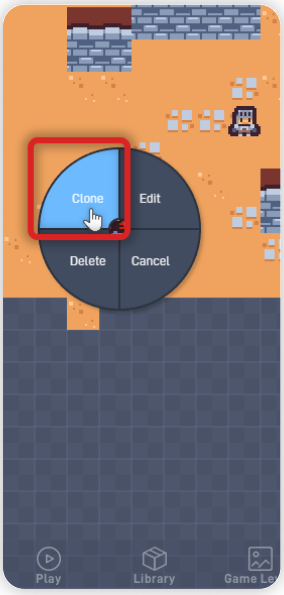


Step 5

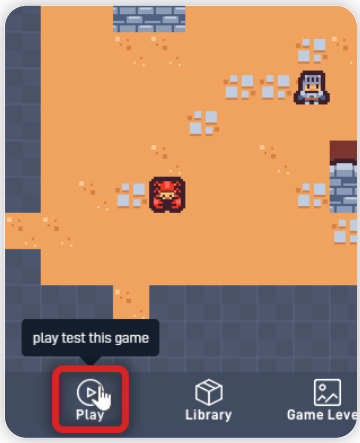
Add more Enemy Objects throughout the Level

Click on the "Enemy" object and select "Clone" from the circle menu.
Click throughout the editor area to add more enemies.
Feel free to add objects outside the game's visible area, as the game view will scroll along with the Player.

Once you finish cloning the objects, click "Done Cloning" on the bottom toolbar.



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



When in play mode:
- Each Enemy object picks a Random Rotation and moves in that Direction.

- Upon touching a wall, or after 18 frames of moving, the Enemy object turns around and moves in the opposite direction.

- The Player can’t move through the Enemy objects (because they are identified as a “Wall” object due to their Parent object).

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 Enemy Sprite rotation doesn’t match the direction it is moving,** ensure its “Forward Direction” is set to “down”; *(Step 4)*
- **If the Enemy object isn’t rotating after colliding with a Wall,** ensure that on its logic, the Collision behavior is set to “Wall” *(Step 2)*, and all your “other Wall” objects have their Parent set to the “Wall” object; *(Step 5)*
- **If the Player can move through the Enemy object,** ensure the Enemy object has its Parent set to “Wall”; *(Step 4)*
- **If the Player object gets pushed around when colliding with other objects,** ensure that on its Physics tab, “movable” is not selected but “is solid” is selected; *(Step 3)*

Top Down Adventure - Part 1

Nice work!

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